

---

# **Bcome Guides**

*Release 2.0.0*

**webzakimbo**

**Oct 18, 2020**



<b>1</b>	<b>EC2</b>	<b>3</b>
<b>2</b>	<b>GCP Service Account</b>	<b>5</b>
<b>3</b>	<b>GCP OAuth 2.0</b>	<b>7</b>
<b>4</b>	<b>Static</b>	<b>9</b>
<b>5</b>	<b>Single Namespace</b>	<b>11</b>
<b>6</b>	<b>Multi-Namespace</b>	<b>13</b>
<b>7</b>	<b>GCP Multi-Network</b>	<b>17</b>
<b>8</b>	<b>AWS Multi-Network</b>	<b>21</b>
<b>9</b>	<b>Alternative Namespace Views</b>	<b>27</b>
<b>10</b>	<b>Merging Clouds</b>	<b>29</b>
<b>11</b>	<b>On-premise</b>	<b>35</b>
<b>12</b>	<b>Static-Cloud</b>	<b>37</b>
<b>13</b>	<b>Hybrid Static Cloud</b>	<b>39</b>
<b>14</b>	<b>Multi-cloud</b>	<b>43</b>
<b>15</b>	<b>Multi-hybrid-cloud</b>	<b>47</b>
<b>16</b>	<b>Overriding identifiers</b>	<b>53</b>
<b>17</b>	<b>Configuration Inheritance</b>	<b>57</b>
<b>18</b>	<b>Simplest SSH configuration</b>	<b>59</b>
<b>19</b>	<b>Basic SSH Proxying</b>	<b>61</b>
<b>20</b>	<b>Multi-proxying</b>	<b>63</b>

<b>21</b>	<b>Overriding SSH Configuration</b>	<b>67</b>
22	tree	69
23	routes	71
24	run	73
25	ssh	75
26	interactive	77
27	tunnel	79
28	ping	81
29	rsync	83
30	pseudo_tty	85
31	put_str	87
32	get	89
33	registry	91
34	Shortcuts	93
35	External methods	95
36	Internal Methods	97

Welcome to the Bcome Guides.

Here you will find quick-fire examples of the core functionality & configuration options within Bcome.

For full documentation to produce your own Control Panel application, please see the [documentation](#).



This guide demonstrates a basic EC2 driver setup: a single inventory namespace is configured to populate itself with some arbitrary servers.

---

**Note:** For further configuration details, please refer to the [documentation](#).

---

## 1.1 Directory structure

You should have a directory structure as follows:

```
.
├── .aws
│   └── keys
├── bcome
│   └── networks.yml
```

The networks.yml file contains your network configuration, whilst 'keys' contains your AWS access keys.

---

**Note:** For further information on linking AWS accounts, see [Aws Authorization](#).

---

## 1.2 Network Configuration

The networks.yml configuration is simple:

```
---
wbz:
  type: inventory
```

(continues on next page)

(continued from previous page)

```
description: Entire WBZ estate

network:
  type: ec2
  credentials_key: webzakimbo
  provisioning_region: eu-west-1
  filters:
    instance-state-name: running

ssh_settings:
  timeout_in_seconds: 10
  proxy:
    host_lookup: by_bcome_namespace
    namespace: bastion
```

---

**Note:** For a full list of namespace attributes see [namespace attributes](#).

---

## 1.3 Ascii Cast

The following Ascii Cast illustrates the above configuration:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/0kwvSqjdk19N3GYE39ZgHzzWP
```

---



This guide demonstrates a basic GCP driver setup: a single inventory namespace is populated with servers having been authorised via a GCP service account.

For further configuration details, please refer to the [documentation](#).

## 2.1 Directory structure

```
.
├── .gauth
│   └── service-account.json
├── bcome
│   └── networks.yml
```

The networks.yml file contains your network configuration, whilst 'service-account.json' contains your GCP service account credentials.

---

**Note:** For further information on linking GCP accounts, see [GCP Authorization](#).

---

## 2.2 Network Configuration

The networks.yml configuration is simple:

```
---
wbz:
  type: inventory
  description: All my servers in a single namespace
  network:
```

(continues on next page)

(continued from previous page)

```
type: gcp
project: wbznet
zone: europe-west1-b
authentication_scheme: service_account
service_account_credentials: service-account.json
service_scopes:
- https://www.googleapis.com/auth/compute.readonly
- https://www.googleapis.com/auth/cloud-platform
filters: status:running

ssh_settings:
  proxy:
    - host_lookup: by_bcome_namespace
      namespace: bastion
```

---

**Note:** For a full list of namespace attributes see [namespace attributes](#).

---

## 2.3 Ascii Cast

The following Ascii Cast illustrates the above configuration:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/YCWMpROQy70UIpUaiU1OemF39
```

---

This guide demonstrates a basic GCP driver setup: a single inventory namespace is populated with servers having been authorised via GCP OAuth 2.0 ([configure GCP using OAuth 2.0](#)).

For further configuration details, please refer to the [documentation](#).

### 3.1 Directory structure

```
.
├── .gauth
│   └── your-secrets-file.json
├── bcome
│   └── networks.yml
```

The `networks.yml` file contains your network configuration, whilst `'your-secrets-file.json'` contains your OAuth 2.0 application secrets.

---

**Note:** Any user requiring use of your OAuth 2.0 application will need the OAuth 2.0 application secrets.

Bcome will trigger an OAuth 2.0 authentication process with first usage (or should the access tokens returned from the OAuth 2.0 process have expired or been invalidated).

---

**Warning:** Access tokens are saved to the `.gauth` directory, the contents of which should not be added to source control.

### 3.2 Network Configuration

The `networks.yml` configuration is simple:

```
---
wbz:
  type: inventory
  description: Entire WBZ estate

  network:
    type: gcp
    project: wbznet
    zone: europe-west1-b
    authentication_scheme: oauth
    secrets_filename: your-secrets-file.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform
    filters: status:running

  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
      namespace: bastion
```

---

**Note:** For a full list of namespace attributes see [namespace attributes](#).

---

### 3.3 Ascii Cast

The following Ascii Cast illustrates the above configuration:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/iskFuzue4LzAx6LIV9144JGuy
```

---

Where a Static Manifest has been set against a given namespace, Bcome will populate that namespace with servers from the manifest.

---

**Note:** See [static manifests](#) for full documentation.

---

A Static Manifest allows for the declaration of servers local to your client, i.e. on-premise, or remote machines for which you may not have a configured Bcome cloud driver.

In this guide we'll add a single static server into the top-level inventory namespace.

## 4.1 Directory structure

You should have a directory structure as follows:

```
.
├── bcome
│   ├── networks.yml
│   └── static-cache.yml
```

## 4.2 Static Manifest

My static-cache.yml file looks as follows:

```
---
wbz:
- identifier: fserver_a
  internal_ip_address: 192.168.1.50
  local_network: yes
```

(continues on next page)

(continued from previous page)

```
description: Central store
cloud_tags:
  data:
    environment: office
    function: filestore
    group: administrative
```

It declares a single server on my local network.

## 4.3 Network Configuration

My networks.yml configuration is extremely simple: it declares a top-level Inventory namespace, for which no cloud driver has been declared.

```
---
wbz:
  type: inventory
  description: Entire WBZ estate
  network: {}
```

## 4.4 Ascii Cast

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/THHfySR7m6V50VkWKBjw7OSzG
```

---

---

## Single Namespace

---

The most simple Bcome setup is where your servers are loaded into a single namespace.

Let's see how this works for an inventory retrieved from Google Cloud Platform where I have two servers configured - a bastion server and an application server.

For further configuration details, please refer to the [documentation](#).

### 5.1 Tree Hierarchy

Here's the tree hierarchy for this guide - a single inventory containing two servers.



### 5.2 Network Configuration

The network configuration is simple:

```
---
wbz:
  type: inventory
  description: All my servers in a single namespace

  network:
    type: gcp
    project: wbznet
    zone: europe-west1-b
    authentication_scheme: service_account
```

(continues on next page)

(continued from previous page)

```
service_account_credentials: service-account.json
service_scopes:
- https://www.googleapis.com/auth/compute.readonly
- https://www.googleapis.com/auth/cloud-platform
filters: status:running

ssh_settings:
  proxy:
    - host_lookup: by_bcome_namespace
      namespace: bastion
```

### 5.3 Ascii Cast

---

**Note:** To replay this Ascii cast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/VGmOXnE89Qwd7yqQozuWuAIS4
```

---



---

## Multi-Namespace

---

It's best if you segment your infrastructure into namespaces.

Think of a namespace as a particular “view” on a part, or parts of your infrastructure.

In this example, I've an elastic search cluster, an application server, and some management servers set up in GCP.

I've created my namespaces in this instance using `sub-selected inventories`. Please refer to the docs - <https://docs.bcome.com> - for more information on the namespace types available.

---

**Note:** The servers retrieved in this example are tagged in GCP with various labels. These labels are the subject of the filters you'll see in the Network Configuration.

---

### 6.1 Project structure

```
.
├── bcome
│   └── networks.yml
```

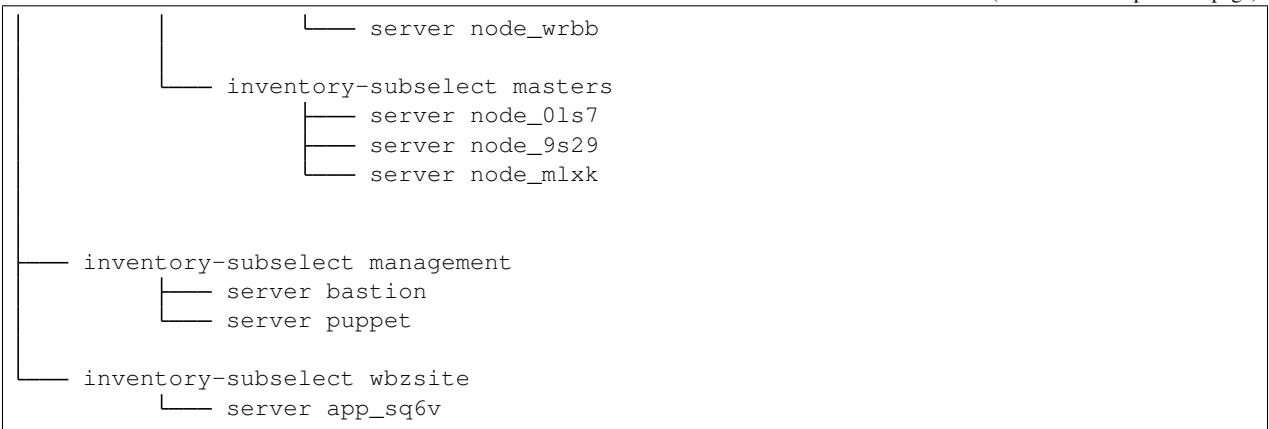
### 6.2 Tree Hierarchy

The tree hierarchy below is generated by invoking Bcome's `tree` command:

```
Namespace tree wbz
├── collection elastic
│   └── inventory-subselect data
│       ├── server node_81rs
│       └── server node_mtk9
```

(continues on next page)

(continued from previous page)



## 6.3 Network Configuration

The following network configuration sets up a multi-namespace views. It organises servers from a single cloud provider, GCP, into namespaces by filtering on the tags set on those servers within GCP.

```

---
wbz:
  type: collection
  description: All my servers in multiple namespaces
  network:
    type: gcp
    project: wbznet
    zone: europe-west1-b
    :authentication_scheme: service_account
    service_account_credentials: service-account.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform

  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: management:bastion

wbz:all_machines:
  hidden: true
  type: inventory
  description: All Production environment
  override_identifier: "prod_net_(.+)"
  network:
    filters: status:running AND labels.environment=prod-net

wbz:management:
  type: inventory-subselect
  subselect_from: all_machines
  description: Operations namespace
  filters:
    by_label:
      group: operations
  
```

(continues on next page)

(continued from previous page)

```
wbz:wbzsite:
  type: inventory-subselect
  subselect_from: all_machines
  description: Frontend wbzsite
  override_identifier: "wbzsite_(.+)"
  filters:
    by_label:
      group: application
      function: frontend-wbzsite

wbz:elastic:
  type: collection
  description: Elastic search cluster

wbz:elastic:data:
  type: inventory-subselect
  description: elastic search data nodes
  subselect_from: all_machines
  override_identifier: "elastic_data_(node_.+)"
  filters:
    by_label:
      division: elastic-search
      function: elastic-data-node

wbz:elastic:masters:
  type: inventory-subselect
  description: elastic search master nodes
  subselect_from: all_machines
  filters:
    by_label:
      division: elastic-search
      function: elastic-master-node
  override_identifier: "elastic_master_(node_.+)"
```

---

**Note:** Always tag your cloud assets if you can.

---

## 6.4 Ascii Cast

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/SNXoHJ1dFMJVtuokSh9V6VY8i
```

---



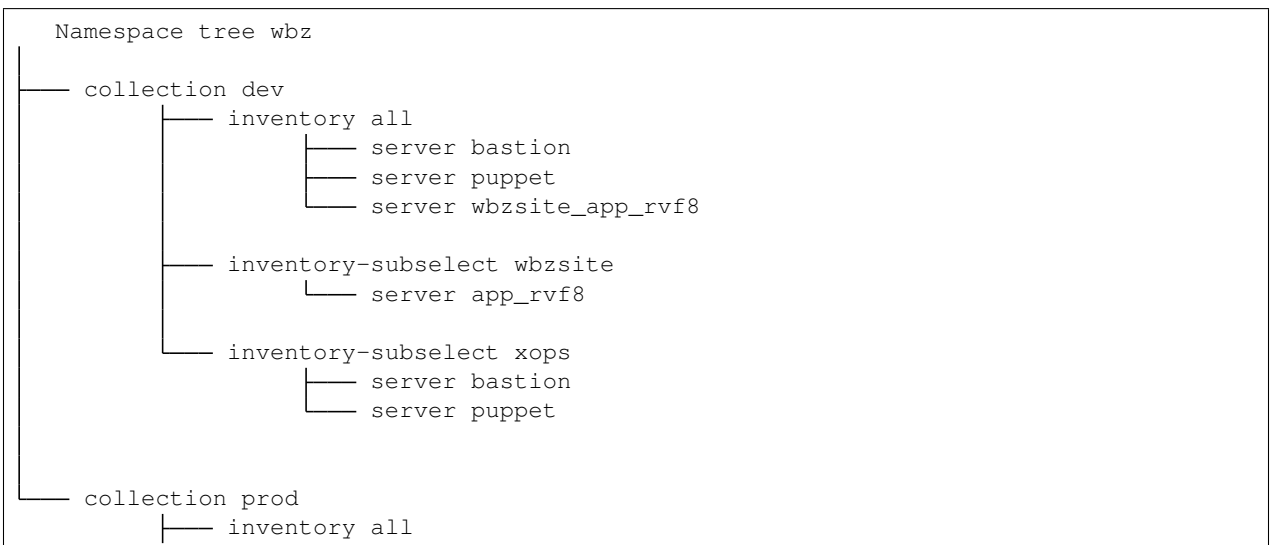
It's likely that your platform is comprised of multiple environments. You can map these environment in your namespaces.

In this example, I've two application environments setup in GCP - "prod" and "dev". Both environments have been built from the same Terraform template - they are identical.

**Note:** For further configuration details, please refer to the [documentation](#).

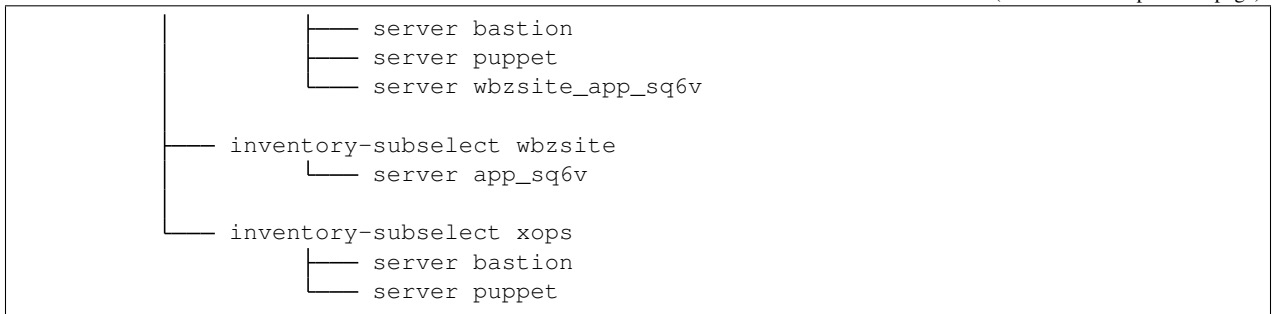
## 7.1 Tree Hierarchy

The tree hierarchy below is generated by invoking Bcome's `tree` command:



(continues on next page)

(continued from previous page)



## 7.2 Network Configuration

Below is an example multi-network GCP configuration: two networks, corresponding to development & production application environments, are configured.

```

---
wbz:
  type: collection
  description: Entire WBZ estate
  ssh_settings: {}

  network:
    type: gcp
    project: wbznet
    authentication_scheme: oauth
    secrets_filename: wbz-net-oauth-secrets.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform

wbz:prod:
  type: collection
  description: GCP Production
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: prod:xops:bastion
  network:
    filters: status:running AND labels.environment=prod-net
    zone: europe-west1-b

wbz:prod:all:
  hidden: false
  type: inventory
  description: All Production environment
  override_identifier: "prod_net_(.+)"

wbz:prod:xops:
  type: inventory-subselect
  subselect_from: prod:all
  description: Operations namespace
  filters:
    by_label:

```

(continues on next page)

(continued from previous page)

```

    group: operations

wbz:prod:wbzsite:
  type: inventory-subselect
  subselect_from: prod:all
  description: Frontend wbzsite
  override_identifier: "wbzsite_(.+)"
  filters:
    by_label:
      group: application
      function: frontend-wbzsite

wbz:dev:
  type: collection
  description: GCP Development
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: dev:xops:bastion
  network:
    filters: status:running AND labels.environment=dev-net
    zone: europe-west1-c

wbz:dev:all:
  hidden: false
  type: inventory
  description: All Development enviornment
  override_identifier: "dev_net_(.+)"

wbz:dev:xops:
  type: inventory-subselect
  subselect_from: dev:all
  description: Operations namespace
  filters:
    by_label:
      group: operations

wbz:dev:wbzsite:
  type: inventory-subselect
  subselect_from: dev:all
  description: Frontend wbzsite
  override_identifier: "wbzsite_(.+)"
  filters:
    by_label:
      group: application
      function: frontend-wbzsite

```

## 7.3 Ascii Cast

The following Ascii cast presents a quick run-through of navigating the namespace configuration.

**Note:** To replay this Ascii cast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

asciinema play <https://asciinema.org/a/gF172t4mFX42djQDdJIxVQtp5>

---



It's likely that your platform is comprised of multiple environments. You can map these environment in your namespaces.

I've two application environments setup in AWS - "prod" and "dev". Both environments have been built from the same Terraform template - they are identical.

---

**Note:** In this example, each Bcome namespace is populated by a separate lookup against EC2 - i.e. each namespace maps to its own Inventory.

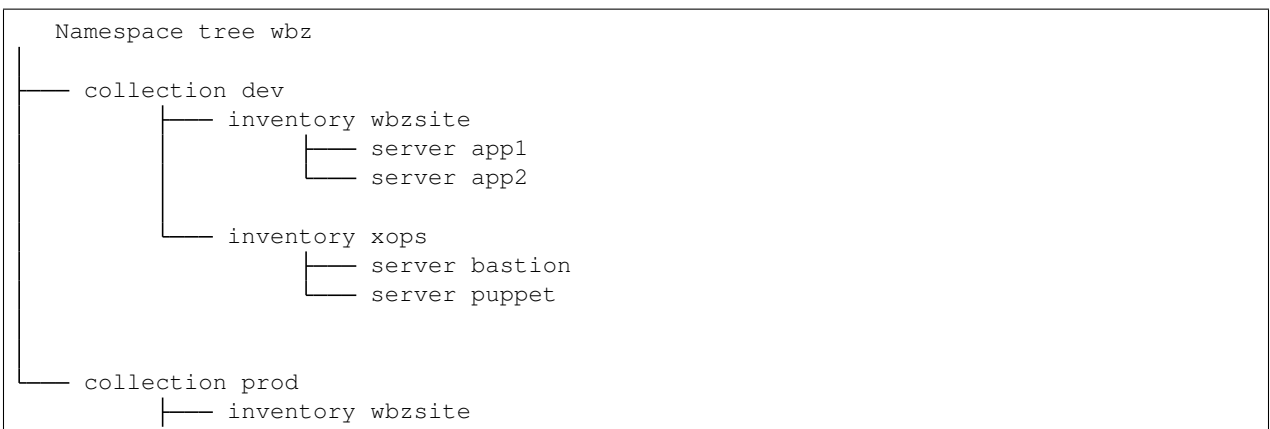
Filter keys map to labels set on servers in EC2.

For further configuration details, please refer to the [documentation](#).

---

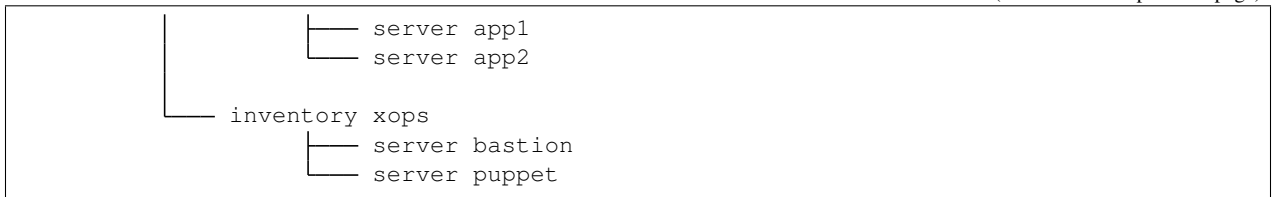
## 8.1 Tree Hierarchy

Take a look the tree hierarchy, generated by invoking Bcome's `tree` command:



(continues on next page)

(continued from previous page)



## 8.2 Network Configuration

Below is an example multi-network AWS configuration: two networks, corresponding to development & production application environments, are configured.

```

---
wbz:
  type: collection
  description: WBZ aws estate
  ssh_settings:
    timeout_in_seconds: 10
  network:
    type: ec2
    credentials_key: webzakimbo
    provisioning_region: eu-west-1
    filters:
      instance-state-name: running

wbz:dev:
  type: collection
  description: All dev environment
  ssh_settings:
    proxy:
      host_lookup: by_bcome_namespace
      namespace: dev:xops:bastion
  network:
    filters:
      tag:stack: dev-net

wbz:dev:xops:
  type: inventory
  description: Operations namespace
  network:
    filters:
      tag:division: "xops"

wbz:dev:wbzsite:
  type: inventory
  description: Frontend wbzsite
  network:
    filters:
      tag:function: "frontend-wbzsite"

wbz:prod:
  type: collection
  description: All prod environment
  ssh_settings:

```

(continues on next page)

(continued from previous page)

```

proxy:
  host_lookup: by_bcome_namespace
  namespace: prod:xops:bastion
network:
  filters:
    tag:stack: prod-net

wbz:prod:xops:
  type: inventory
  description: Operations namespace
  network:
    filters:
      tag:division: "xops"

wbz:prod:wbzsite:
  type: inventory
  description: Frontend wbzsite
  network:
    filters:
      tag:function: "frontend-wbzsite"

```

## 8.3 Ascii Cast

The following AsciiCast presents a quick run-through of navigating the namespace configuration.

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/YQ5oNHDABJ7wjkGvdLKvYKWZj
```

## 8.4 Alternative Network Configuration

The previous example performs four lookups against EC2 (one per inventory). We may reduce the number of lookups by using the `inventory-subselect` namespace type:

```

---
wbz:
  type: collection
  description: WBZ aws estate
  ssh_settings:
    timeout_in_seconds: 10
  network:
    type: ec2
    credentials_key: webzakimbo
    provisioning_region: eu-west-1
    filters:
      instance-state-name: running

wbz:dev:
  type: collection

```

(continues on next page)

(continued from previous page)

```
description: All dev environment
ssh_settings:
  proxy:
    host_lookup: by_bcome_namespace
    namespace: dev:xops:bastion
network:
  filters:
    tag:stack: dev-net

wbz:dev:all:
  type: inventory
  description: all development servers
  hidden: true

wbz:dev:xops:
  type: inventory-subselect
  description: Operations namespace
  subselect_from: dev:all
  filters:
    by_tag:
      division:
        - "xops"

wbz:dev:wbzsite:
  type: inventory-subselect
  description: Frontend wbzsite
  subselect_from: dev:all
  filters:
    by_tag:
      function: "frontend-wbzsite"

wbz:prod:
  type: collection
  description: All prod environment
  ssh_settings:
    proxy:
      host_lookup: by_bcome_namespace
      namespace: prod:xops:bastion
network:
  filters:
    tag:stack: prod-net

wbz:prod:all:
  type: inventory
  description: all production servers
  hidden: true

wbz:prod:xops:
  type: inventory-subselect
  description: Operations namespace
  subselect_from: prod:all
  filters:
    by_tag:
      division:
        - "xops"

wbz:prod:wbzsite:
```

(continues on next page)

(continued from previous page)

```
type: inventory-subselect
description: Frontend wbzsite
subselect_from: prod:all
filters:
  by_tag:
    function: "frontend-wbzsite"
```

The above will result in the exact same namespace configuration.



---

### Alternative Namespace Views

---

You may generate alternative views on the same infrastructure by using the `CONF=` environment variable. This is useful if you need to provide different views for different teams, without having to generate whole new installations.

See [Alternative Configuration with CONF=](#) for full configuration details & usage instructions.

The following AsciiCast demonstrates how the same infrastructure may be visualised differently by specifying alternative network configuration using the `CONF=` environment variable:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/xEBnvNuFsWMuBDLgjEzdRCv1T
```

---





# CHAPTER 10

---

## Merging Clouds

---

Bcome lets you create namespaces across disparate clouds. This example shows a simple merged AWS EC2 & GCP inventory, where application servers from both clouds are placed into one inventory.

You may interact with merged inventories in the usual manner: via the console, from the terminal, or from your custom orchestration scripts.

The point of this guide is to demonstrate how simple this connectivity can be to set up.

---

**Note:** You may merge any inventories - irrespective of their origins.

---

### 10.1 Network Configuration

Below is my network configuration - my networks.yml configuration file. It defines two namespaces - one each per GCP & EC2, and within each I have a namespace named 'wbzsite' containing application servers, and a 'jump' namespace containing a jump host through which ingress into each respective network is achieved.

I then define an inventory of type `inventory-merge` which I name `multicloud_app` which provides a view on all my application servers from both cloud providers.

```
---
wbz:
  type: collection
  description: Entire WBZ estate

wbz:aws:
  type: collection
  description: WBZ aws estate

ssh_settings:
  proxy:
    host_lookup: by_bcome_namespace
```

(continues on next page)

```

    namespace: aws:jump:bastion

network:
  type: ec2
  credentials_key: webzakimbo
  provisioning_region: eu-west-1
  filters:
    instance-state-name: running

wbz:aws:all:
  type: inventory
  description: all production servers
  hidden: true

network:
  filters:
    tag:stack: prod-net

wbz:aws:jump:
  type: inventory-subselect
  description: Operations namespace
  subselect_from: aws:all
  ssh_settings:
    proxy: []
  filters:
    by_tag:
      division: "xops"
      function: "bastion"

wbz:aws:wbzsite:
  type: inventory-subselect
  description: Frontend wbzsite
  subselect_from: aws:all
  filters:
    by_tag:
      function: "frontend-wbzsite"

wbz:gcp:
  type: collection
  description: WBZ gcp estate
  network:
    type: gcp
    project: wbznet
    authentication_scheme: oauth
    secrets_filename: wbz-net-oauth-secrets.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform
    filters: status:running AND labels.environment=prod-net
    zone: europe-west1-b

  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: gcp:jump:bastion

wbz:gcp:all:

```

(continues on next page)

(continued from previous page)

```

hidden: true
type: inventory
description: All Production environment

wbz:gcp:jump:
  type: inventory-subselect
  subselect_from: gcp:all
  description: Public machines
  ssh_settings:
    proxy: []
  filters:
    by_label:
      group: operations
      function:
        - bastion

wbz:gcp:wbzsite:
  type: inventory-subselect
  subselect_from: gcp:all
  description: Frontend wbzsite
  filters:
    by_label:
      group: application
      function: frontend-wbzsite

wbz:multicloud_app:
  type: inventory-merge
  description: All application servers from AWS & GCP
  contributors:
    - gcp:wbzsite
    - aws:wbzsite

```

## 10.2 Tree Hierarchy

Take a look the tree hierarchy, generated by invoking Bcome's `tree` command:

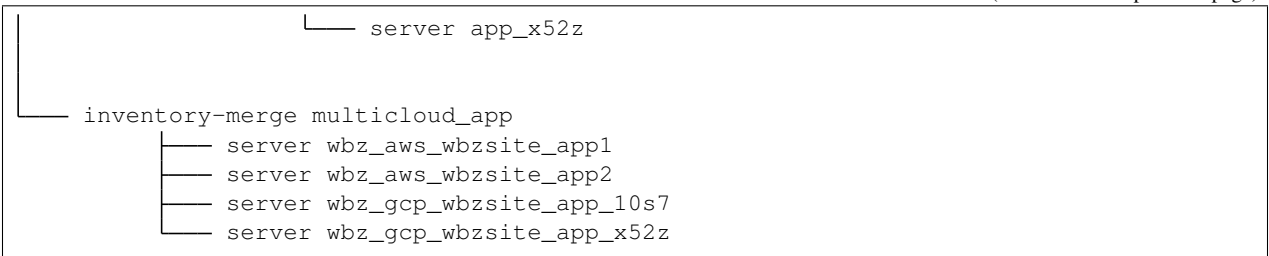
```

Namespace tree wbz
├── collection aws
│   ├── inventory-subselect jump
│   │   └── server bastion
│   └── inventory-subselect wbzsite
│       ├── server appl
│       └── server app2
├── collection gcp
│   ├── inventory-subselect jump
│   │   └── server bastion
│   └── inventory-subselect wbzsite
│       └── server app_10s7

```

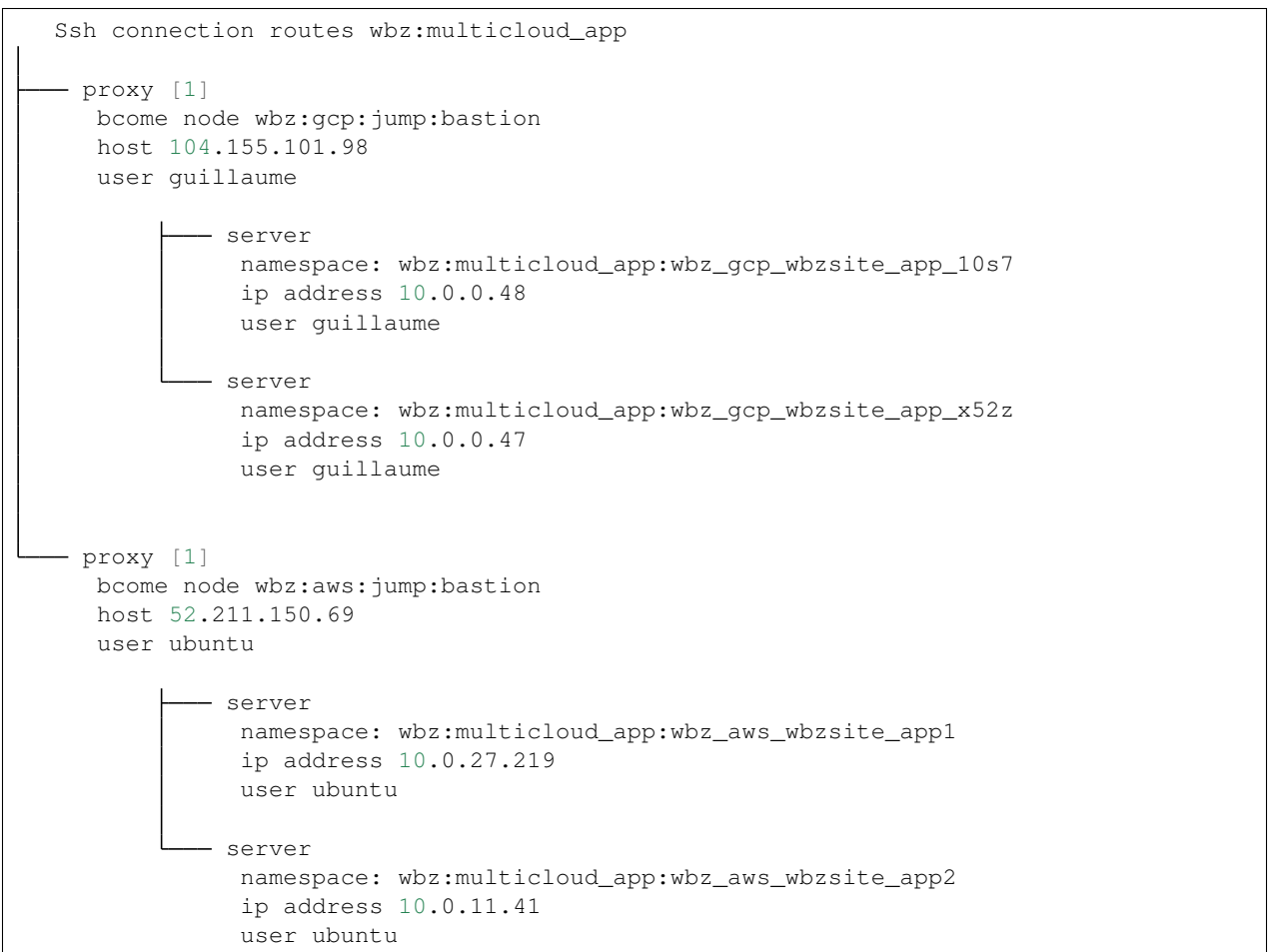
(continues on next page)

(continued from previous page)



## 10.3 Routes

Take a look at the `routes` output for my `multicloud` namespace - ingress to each respective cloud is achieved through different jump hosts, each origin cloud maintaining its own SSH configuration:



## 10.4 Ascii Cast

The following AsciiCast presents a quick demonstration of connectivity to our multi-cloud merged inventory:

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/vW24SCPt8ZXpZ9Hq0BurPZ70F
```

---



Where a Static Manifest has been set against a given namespace, Bcome will populate that namespace with servers from the manifest.

In this simple example, we add a single server into the top-level inventory namespace.

---

**Note:** See [static manifests](#) for full documentation.

---

## 11.1 Directory structure

```
.
├── bcome
│   ├── networks.yml
│   └── static-cache.yml
```

## 11.2 Static Cache manifest

```
---
wbz:
- identifier: fserver_a
  internal_ip_address: 192.168.1.50
  local_network: yes
  description: Central store
  cloud_tags:
    data:
      environment: office
      function: filestore
      group: administrative
```

## 11.3 Network Configuration

The networks.yml file is very simple - there is no need to specify a cloud driver as Bcome will default to loading in the declared Static Cache.

```
----  
wbz:  
  type: inventory  
  description: Entire WBZ estate  
  network: {}
```

## 11.4 Ascii Cast

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/qsdv1GamV0UX36OK8wK1BHBBK
```

---



Where a Static Manifest has been set against a given namespace, Bcome will populate that namespace with servers from the manifest (see: [static manifests](#)).

As well as using this pattern to configure on-premise infrastructure, you may add in remote infrastructure for which you may not necessarily have a Bcome driver installed.

In this example, we populate an inventory with three remote servers from a static manifest.

## 12.1 Directory structure

```
.
├── bcome
│   ├── networks.yml
│   └── static-cache.yml
```

## 12.2 Static Cache manifest

The static-cache.yml configuration below defines three remote servers.

```
---
wbz:
- identifier: bastion
  internal_ip_address: 10.2.0.2
  public_ip_address: 35.205.188.41
  description: GCP server - prod-net-bastion
  cloud_tags:
    data:
      environment: prod-net
      function: bastion
      group: operations
```

(continues on next page)

(continued from previous page)

```
- identifier: puppet
  internal_ip_address: 10.0.0.10
  description: GCP server - prod-net-puppet
  cloud_tags:
    data:
      environment: prod-net
      function: puppet
      group: operations
- identifier: wbzsite_app_s27x
  internal_ip_address: 10.0.0.2
  description: GCP server - prod-net-wbzsite-app-s27x
  cloud_tags:
    data:
      group: application
      environment: prod-net
      function: frontend-wbzsite
```

## 12.3 Network Configuration

The networks.yml configuration remains simple - there is no need to specify a cloud driver, the system will default to populating the inventory from the Static Cache.

```
---
wbz:
  type: inventory
  description: Entire WBZ estate

  network: {}

  ssh_settings:
    timeout_in_seconds: 10
  proxy:
    host_lookup: by_bcome_namespace
    namespace: bastion
```

## 12.4 Ascii Cast

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/MTctnlcAnAWdGt8nG0r1N7mjp
```

Perhaps you have on-premise & remote servers that you wish to use within the same installation.

In this example, we'll populate one namespace with an on-premise fileserver, and another with a few servers from GCP. As a final step, a merged inventory is created demonstrating how to interact with all the servers at once.

### 13.1 Directory structure

```
.
├── bcome
│   ├── networks.yml
│   └── static-cache.yml
```

### 13.2 Static Cache Manifest

Here we define a single local server:

```
---
wbz:on_premise:
- identifier: fserver_a
  internal_ip_address: 192.168.1.50
  local_network: yes
  description: Central store
  cloud_tags:
    data:
      environment: office
      function: filestore
      group: administrative
```

## 13.3 Network Configuration

The network.yml configuration specifies three inventories: One populated from the cloud, a second populated from a static cache, and a third merging cloud & static.

```

---
wbz:
  type: collection
  description: Entire WBZ estate

wbz:on_premise:
  type: inventory
  description: on-premise infrastructure

wbz:gcp:
  type: inventory
  description: GCP machines
  network:
    type: gcp
    project: wbznet
    zone: europe-west1-b
    authentication_scheme: oauth
    secrets_filename: wbz-net-oauth-secrets.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform
    filters: status:running

  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: gcp:bastion

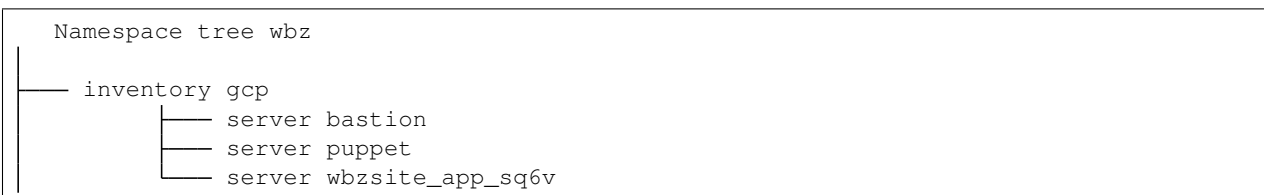
  override_identifier: "[a-z]*_[a-z]*_(.+)"

wbz:hybrid:
  type: inventory-merge
  description: GCP & on-premise infrastructure
  contributors:
    - gcp
    - on_premise

```

## 13.4 Tree Hierarchy

Illustrated below is the installation's tree structure. The “gcp” namespace contains servers populated from Google Cloud Platform. The “on\_premise” is a local fileservers, whilst the “hybrid” namespace merges both allowing orchestration of all at the same time.



(continues on next page)

(continued from previous page)

```

inventory-merge hybrid
├── server wbz_gcp_bastion
├── server wbz_gcp_puppet
├── server wbz_gcp_wbzsite_app_sq6v
├── server wbz_on_premise_fserver_a
└── inventory on_premise
    └── server fserver_a

```

**Note:** Note how the merged inventory retains the full server identifiers. This prevents name conflicts when similar inventories are used as contributors to a merge.

## 13.5 SSH Routing Tree

The following routing tree (generated using Bcome's `routes` command) illustrates how the system will connect to the servers within it.

```

Ssh connection routes wbz
├── server
│   ├── namespace: wbz:on_premise:fserver_a
│   ├── ip address 192.168.1.50
│   └── user guillaume
├── proxy [1]
│   ├── bcome node wbz:gcp:bastion
│   ├── host 104.155.101.98
│   └── user guillaume
│
│   ├── server
│   │   ├── namespace: wbz:gcp:bastion
│   │   ├── ip address 10.2.0.2
│   │   └── user guillaume
│   ├── server
│   │   ├── namespace: wbz:gcp:puppet
│   │   ├── ip address 10.0.0.10
│   │   └── user guillaume
│   └── server
│       ├── namespace: wbz:gcp:wbzsite_app_sq6v
│       ├── ip address 10.0.0.2
│       └── user guillaume

```

## 13.6 Ascii Cast

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

asciinema play <https://asciinema.org/a/HJWt7HSZCLnth823FhyVcje85>

---

Bcome allows for interacting with servers from multiple clouds at the same time.

This guide demonstrates a simple AWS & GCP integration, where each cloud is used to populate an inventory, and then both used as contributors to populate a merged (multi cloud) inventory.

---

**Note:** A multi-cloud inventory is no different to any other: it may be interacted with through the console, or programmatically from an orchestration script.

---

For documentation on linking AWS accounts, see: [Aws Authorization](#).

For documentation on linking GCP accounts, see: [GCP Authorization](#)

## 14.1 Directory structure

You should have a directory structure as follows:

```
.
├── .aws
│   └── keys
├── .gauth
│   └── service-account.json
├── bcome
│   └── networks.yml
```

## 14.2 Network Configuration

```
---
wbz :
```

(continues on next page)

```

type: collection
description: Entire WBZ estate

wbz:aws:
type: inventory
description: AWS machines
network:
  type: ec2
  credentials_key: webzakimbo
  provisioning_region: eu-west-1
  filters:
    instance-state-name: running

  ssh_settings:
    timeout_in_seconds: 10
    proxy:
      host_lookup: by_bcome_namespace
      namespace: aws:bastion

  override_identifier: "[a-z]*_[a-z]*[a-z]*_(.+)"

wbz:gcp:
type: inventory
description: GCP machines
network:
  type: gcp
  project: wbznet
  zone: europe-west1-b
  authentication_scheme: service_account
  service_account_credentials: service-account.json
  service_scopes:
    - https://www.googleapis.com/auth/compute.readonly
    - https://www.googleapis.com/auth/cloud-platform
  filters: status:running

  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
      namespace: gcp:bastion

  override_identifier: "[a-z]*_[a-z]*_(.+)"

wbz:multicloud:
type: inventory-merge
description: GCP & AWS
contributors:
  - gcp
  - aws

```

## 14.3 Tree Hierarchy

Illustrated below is the installation's tree structure.

The “gcp” namespace contains servers populated from Google Cloud Platform. The “aws” namespace contains servers



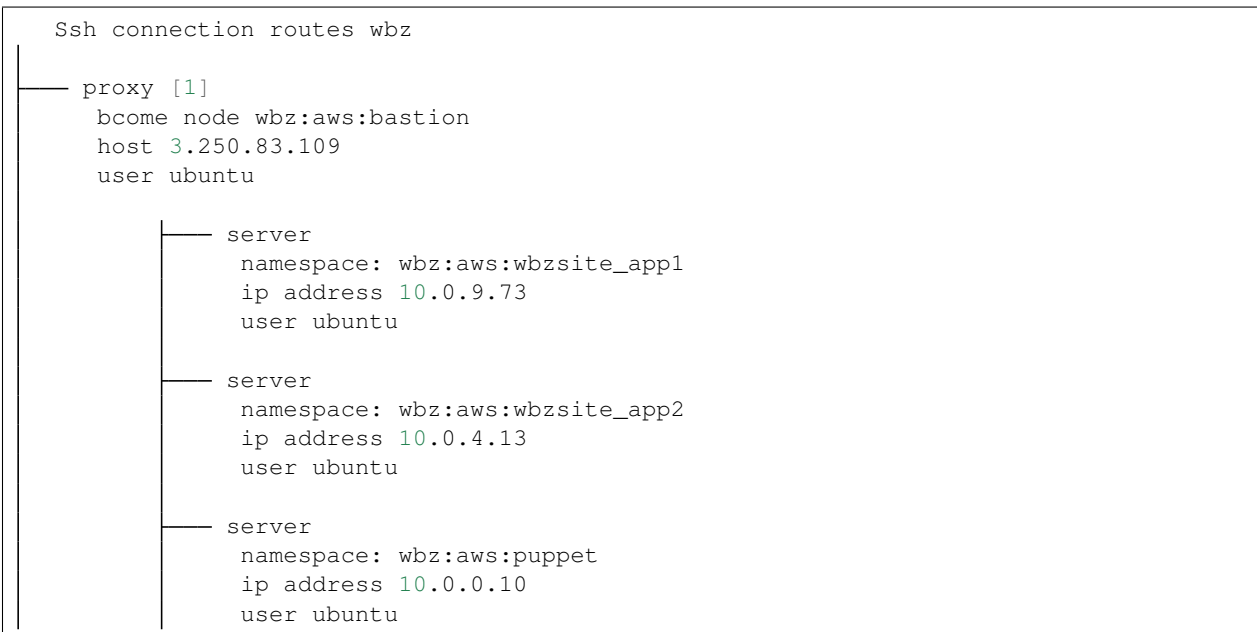
populated from Amazon Web Services. The “multicloud” namespace merges them both.



**Note:** Note how the merged inventory retains the full server identifiers. This prevents name conflicts when similar inventories are used as contributors to a merge.

## 14.4 SSH Routing tree

The routing below illustrates the two connection pathways that Bcome will use when interacting with the servers within the installation.



(continues on next page)

(continued from previous page)

```
server
  namespace: wbz:aws:bastion
  ip address 10.0.35.208
  user ubuntu

proxy [1]
  bcome node wbz:gcp:bastion
  host 104.155.101.98
  user guillaume

server
  namespace: wbz:gcp:bastion
  ip address 10.2.0.2
  user guillaume

server
  namespace: wbz:gcp:puppet
  ip address 10.0.0.10
  user guillaume

server
  namespace: wbz:gcp:wbzsite_app_sq6v
  ip address 10.0.0.2
  user guillaume
```

## 14.5 Ascii Cast

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/6o3aRMAMZ10Kd7if3Bfr3rDqb
```

---

# CHAPTER 15

---

## Multi-hybrid-cloud

---

Bcome allows for interacting with servers from multiple clouds, and on-premise infrastructure at the same time.

This guide demonstrates a simple AWS, GCP and on-premise integration, where each source is used to populate an inventory, and then all three used as contributors to populate a merged (multi-hybrid-cloud) inventory.

---

**Note:** A multi-hybrid-cloud inventory is no different to any other: it may be interacted with through the console, or programmatically from an orchestration script.

---

### 15.1 Directory structure

```
.
├── .aws
│   └── keys
├── .gauth
│   └── service-account.json
├── bcome
│   ├── networks.yml
│   └── static-cache.yml
```

### 15.2 Network Configuration

```
---
wbz:
  type: collection
  description: Entire WBZ estate

wbz:on_premise:
```

(continues on next page)

```
type: inventory
description: on-premise infrastructure

wbz:aws:
type: inventory
description: AWS machines
network:
  type: ec2
  credentials_key: webzakimbo
  provisioning_region: eu-west-1
  filters:
    instance-state-name: running

ssh_settings:
  timeout_in_seconds: 10
  proxy:
    host_lookup: by_bcome_namespace
    namespace: aws:bastion

override_identifier: "[a-z]*_[a-z]*[a-z]*_(.+)"

wbz:gcp:
type: inventory
description: GCP machines
network:
  type: gcp
  project: wbznet
  zone: europe-west1-b
  authentication_scheme: oauth
  secrets_filename: wbz-net-oauth-secrets.json
  service_scopes:
    - https://www.googleapis.com/auth/compute.readonly
    - https://www.googleapis.com/auth/cloud-platform
  filters: status:running

ssh_settings:
  proxy:
    - host_lookup: by_bcome_namespace
      namespace: gcp:bastion

override_identifier: "[a-z]*_[a-z]*_(.+)"

wbz:hybrid:
type: inventory-merge
description: GCP & on-premise infrastructure
contributors:
  - gcp
  - aws
  - on_premise
```

## 15.3 Static Cache Manifest

```

---
wbz:on_premise:
- identifier: fileserver_a
  internal_ip_address: 192.168.0.24
  local_network: yes
  description: Office filestore
  cloud_tags:
    data:
      environment: office
      function: filestore
      group: administrative

```

## 15.4 Tree Hierarchy

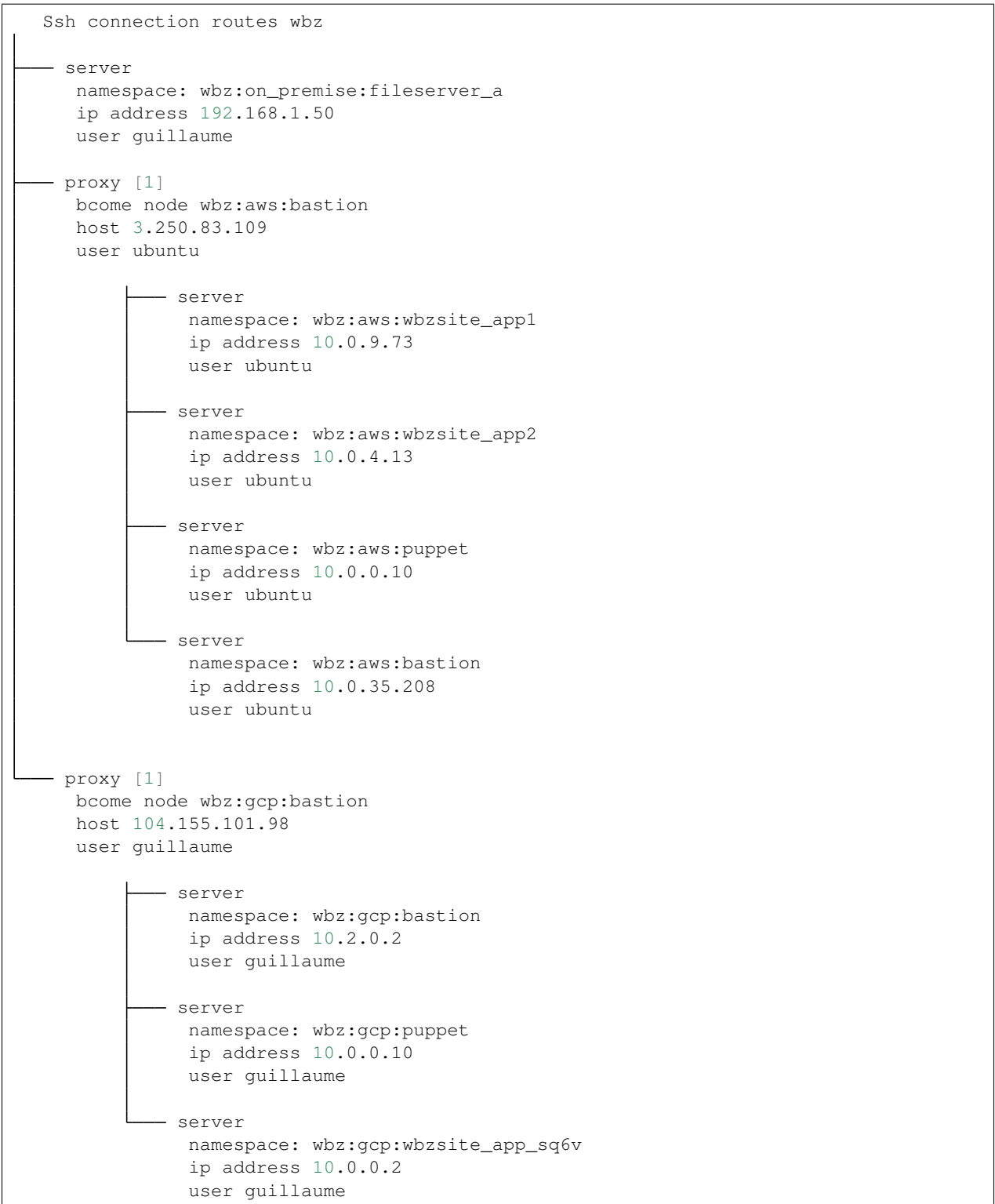
Illustrated below is the installation's tree structure.

The “gcp” namespace contains servers populated from Google Cloud Platform. The “aws” namespace contains servers populated from Amazon Web Services. The “on\_premise” namespaces defines a local file server. The “hybrid” namespace merges all three.



**Note:** Note how the merged inventory retains the full server identifiers. This prevents name conflicts when similar inventories are used as contributors to a merge.

## 15.5 SSH Routing tree



## 15.6 Ascii Cast

The following AsciiCast presents a quick run-through of navigating the namespace configuration.

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/0WfGGYxUpR5gm2heeWFK4SpvJ
```

---





---

## Overriding identifiers

---

Within a given inventory you may provide a regular expression to alter the identifiers (i.e. the names) of your servers as they appear and are referenced within Bcome.

This is useful when you have server names that include metadata that you want to strip within Bcome.

The regular expression must match the server name, and include a single selector that is used to replace the original name.

In the following example, I have setup two GCP inventories. Both return the same servers, but one inventory alters their names with an override removing the prefix “**prod\_net\_**”.

---

**Note:** For further configuration details, please refer to the [documentation](#).

---

### 16.1 Tree Hierarchy

The tree hierarchy below is generated by invoking Bcome’s `tree` command:

```
Namespace tree wbz
├── collection gcp
│   ├── inventory not_overriden
│   │   ├── server prod_net_bastion
│   │   ├── server prod_net_puppet
│   │   └── server prod_net_wbzsite_app_sq6v
│   └── inventory overridden
│       ├── server bastion
│       ├── server puppet
│       └── server wbzsite_app_sq6v
```

## 16.2 Network Configuration

```
---
wbz:
  type: collection
  description: Entire WBZ estate
  ssh_settings: {}

wbz:gcp:
  type: collection
  description: WBZ gcp estate
  network:
    type: gcp
    project: wbznet
    authentication_scheme: oauth
    secrets_filename: wbz-net-oauth-secrets.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform

wbz:gcp:not_overriden:
  type: inventory
  description: GCP Production
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: gcp:not_overriden:prod_net_bastion
  network:
    filters: status:running AND labels.environment=prod-net
    zone: europe-west1-b

wbz:gcp:overriden:
  type: inventory
  description: GCP Production
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: gcp:overriden:bastion
  network:
    filters: status:running AND labels.environment=prod-net
    zone: europe-west1-b

  override_identifier: "prod_net_(.+)"
```

---

**Note:** You may override identifiers within namespaces of type `inventory`, or `inventory-subselect`

Any reference to an overridden server uses the overridden name. See the proxy referenced by “`gcp:overriden:bastion`” above as an example.

---

## 16.3 Ascii Cast

The following AsciiCast presents a quick run-through of navigating the namespace configuration.

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/l28TSXqlLjQCF9trrbtvO5KSL
```

---



---

## Configuration Inheritance

---

SSH and networking configuration is inherited by child namespaces at which point it may be overridden (see: [configuration inheritance](#)).

Here we show a simple two-network GCP setup, where both inherit part of their networking configuration from their parent namespace.

### 17.1 Network Configuration

Below can be seen an example network.yml configuration. The “gcp:prod:” and “gcp:dev:” namespaces both inherit elements of their network configuration from the parent “gcp:” collection, then override that configuration with their own authentication scheme (a service account for one, OAuth 2.0 for the other) and their own network filters.

```
---
wbz:
  type: collection
  description: Entire WBZ estate

wbz:gcp:
  type: collection
  description: WBZ gcp estate
  network:
    type: gcp
    project: wbznet
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform

wbz:gcp:prod:
  type: inventory
  description: GCP Production
  ssh_settings:
    proxy:
```

(continues on next page)

(continued from previous page)

```
- host_lookup: by_bcome_namespace
  namespace: gcp:prod:bastion
network:
  filters: status:running AND labels.environment=prod-net
  authentication_scheme: oauth
  secrets_filename: wbz-net-oauth-secrets.json
  zone: europe-west1-b

wbz:gcp:dev:
  type: inventory
  description: GCP Production
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: gcp:dev:bastion
  network:
    filters: status:running AND labels.environment=dev-net
    authentication_scheme: service_account
    service_account_credentials: service-account.json
    zone: europe-west1-c
```

Any SSH or network configuration may be defined in this way.

### 17.1.1 Ascii Cast

The following Ascii Cast illustrates the above configuration:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/C2m3rAOEHTp72RrNSVetkGkYa
```

---

---

## Simplest SSH configuration

---

The simplest SSH configuration is to define an empty `ssh_settings` block, or to leave it out entirely.

In this case, Bcome will fallback to default system SSH settings:

- Your local user (i.e. the system user running the Bcome process) will be used as the SSH user
- You will not be able to proxy SSH connections, all connections will be direct.

---

**Note:** In all cases - whether SSH is invoked programmatically or otherwise - Bcome will defer to your local ssh-agent for your SSH keys.

Make sure that your ssh-agent is running.

---

### 18.1 Example configuration

The `networks.yml` example below - that of single GCP inventory, returning a single server - demonstrates this configuration:

```
---
wbz:
  type: inventory
  description: Entire WBZ estate

network:
  type: gcp
  project: wbznet
  zone: europe-west1-b
  authentication_scheme: oauth
  secrets_filename: wbz-net-oauth-secrets.json
  service_scopes:
  - https://www.googleapis.com/auth/compute.readonly
  - https://www.googleapis.com/auth/cloud-platform
```

(continues on next page)

(continued from previous page)

```
filters: status:running AND labels.function:bastion
ssh_settings: {}
```

My local user is guillaume, and I have ssh keys added to my agent. See how I may interact with server in the Inventory in the AsciiCast below:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/1KJiA2r4GoxQdtKuVSFsjqeMb
```

---

See the Bcome documentation for more detailed configuration options: [SSH Attributes Configuration](#).



# CHAPTER 19

---

## Basic SSH Proxying

---

If you connect to your machines via an intermediary, then you will need to include a Proxy host in your SSH configuration.

**Note:** In all cases - whether SSH is invoked programmatically or otherwise - Bcome will defer to your local ssh-agent for your SSH keys.

Make sure that your ssh-agent is running and that all keys in play have been added.

---

### 19.1 Example configuration

The networks.yml configuration below defines two inventories: one containing a proxy server, and the other containing servers that may only be connected to via the proxy.

```
---
wbz:
  type: collection
  description: WBZ gcp estate
  network:
    type: gcp
    project: wbznet
    zone: europe-west1-b
    authentication_scheme: service_account
    service_account_credentials: service-account.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform

wbz:proxies:
  type: inventory
  description: ssh proxies
```

(continues on next page)

(continued from previous page)

```
    override_identifier: "prod_net_(.+)"
    network:
      filters: status:running AND labels.function=bastion AND labels.environment=prod-
↳net
wbz:servers:
  type: inventory
  description: Servers
  network:
    filters: status:running AND labels.environment=prod-net AND NOT labels.
↳function=bastion
  override_identifier: "prod_net_(.+)"
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: proxies:bastion
```

The 'proxies' inventory contains a single server named 'bastion' that the 'servers' inventory machines are configured above to use as their proxy.

My local user is guillaume, and I have ssh keys added to my agent.

The AsciiCast below demonstrates my configuration:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/Z8wHFA8DwYYHiKaG1oh7ZYenS
```

---

See the Bcome documentation for more detailed & alternative proxy configuration options: [SSH Proxy Attributes Configuration](#).

If you connect to your machines via an intermediary, then you will need to include a Proxy host in your SSH configuration.

This guide expands on the *Basic SSH Proxying* guide to demonstrate how multiple proxies - i.e. a chain of proxies - may be configured.

### 20.1 Example configuration

The `networks.yml` configuration below defines three inventories: one containing a public-facing proxy server, the second a proxy server installed intra-network (and accessible only from the first), whilst the third inventory defines servers reachable by proxying via both proxy servers.

```
---
wbz:
  type: collection
  description: WBZ gcp estate
  network:
    type: gcp
    project: wbznet
    zone: europe-west1-b
    :authentication_scheme: service_account
    service_account_credentials: service-account.json
    service_scopes:
      - https://www.googleapis.com/auth/compute.readonly
      - https://www.googleapis.com/auth/cloud-platform

wbz:public_proxies:
  type: inventory
  description: public ssh proxies
  override_identifier: "prod_net_(.+)"
  network:
    filters: status:running AND labels.function=bastion AND labels.environment=prod-
    net
```

(continues on next page)

(continued from previous page)

```

wbz:private_proxies:
  type: inventory
  description: private ssh proxies
  override_identifier: "prod_net_(.+)"
  network:
    filters: status:running AND labels.function=internal-bastion AND labels.
    ↪environment=prod-net
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: public_proxies:bastion

wbz:servers:
  type: inventory
  description: Servers accessible via two proxy hops
  network:
    filters: status:running AND labels.environment=prod-net AND NOT (labels.
    ↪function=bastion OR labels.function=internal-bastion)
  override_identifier: "prod_net_(.+)"
  ssh_settings:
    proxy:
      - host_lookup: by_bcome_namespace
        namespace: public_proxies:bastion
      - host_lookup: by_bcome_namespace
        namespace: private_proxies:internal_jump

```

My local user is guillaume, and I have ssh keys added to my agent.

---

**Note:** The proxy block in your ssh\_settings is an array of proxies: you may define as many as you like.

---

## 20.2 Routes

Bcome's routes command will result in the following for the above configuration:

```

Ssh connection routes wbz
├── server
│   namespace: wbz:public_proxies:bastion
│   ip address 104.155.101.98
│   user guillaume
├── proxy [1]
│   bcome node wbz:public_proxies:bastion
│   host 104.155.101.98
│   user guillaume
│   └── proxy [2]
│       bcome node wbz:private_proxies:internal_jump
│       host 10.0.33.2
│       user guillaume
└── server

```

(continues on next page)

(continued from previous page)

```
namespace: wbz:servers:puppet
ip address 10.0.0.10
user guillaume

server
namespace: wbz:servers:wbzsite_app_sq6v
ip address 10.0.0.2
user guillaume
```

The AsciiCast below demonstrates my configuration:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/nPKMiZ6fyum56kHAWswg6ywXO
```

---

See the Bcome documentation for more detailed & alternative proxy configuration options: [SSH Proxy Attributes Configuration](#).



---

## Overriding SSH Configuration

---

**Note:** For detailed documentation on configuration overrides, please refer to the documentation:

[Overriding SSH configuration with 'ME' env variable](#)

[Overriding SSH configuration with me.yml configuration file](#)

---

The following network configuration pulls down some servers from AWS EC2 into a single inventory:

```
---
wbz:
  type: inventory
  description: Entire WBZ estate

  network:
    type: ec2
    credentials_key: webzakimbo
    provisioning_region: eu-west-1
    filters:
      instance-state-name: running

  ssh_settings:
    timeout_in_seconds: 10
    proxy:
      host_lookup: by_bcome_namespace
      namespace: bastion
```

My local terminal user (and so default ssh username) is `guillaume`, yet my EC2 machines have not yet been bootstrapped and expect a username of `ubuntu`.

Rather than hardcoding the required username in my `networks.yml` configuration (see: [SSH Attributes Configuration](#)) I can create an override file and reference it when I make calls to `bcome`. I can also save this override file to my configuration directory as `me.yml` so that is loaded automatically.

See the following AsciiCast for a demonstration:

**Note:** To replay this Ascicast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/ydMEvJaozNGl9NtoqZImtbwxE
```

---



## CHAPTER 22

---

### tree

---

The `tree` command returns a visualisation of your Bcome namespace configuration.

The following AsciiCast demonstrates its usage.

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/8OKWT2N2wDLJXGTuU0047QcUC
```

---

See the Bcome documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



---

### routes

---

The `routes` command displays the pathways your SSH connections will take to reach your servers.

The following AsciiCast demonstrates its usage:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/pQRwqdJ50Uukx7GWcgIr530J
```

---

See the `Bcome` documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



---

### run

---

The `run` command is used to execute commands against machines in targetted namespaces.

The following AsciiCast demonstrates the usage of this command:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/gjfvLjaCaSMxpF2NkvutPdUER
```

---

Like all Bcome commands, ‘run’ is invoked against all machines in your target namespace, with all connection handling & network discovery handled by the framework. See ‘run’ in action against a multi-cloud AWS & GCP inventory:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/SMCo90p0nKGDTdjZnFA28pom1
```

---

See the Bcome documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



## CHAPTER 25

---

ssh

---

The `ssh` command is used to initiate an SSH session with a target server.

The following AsciiCast demonstrates the usage of this command:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/BsYYu4FWjP2RDj10cgIg3Oyyf
```

---

See the `Bcome` documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).





## CHAPTER 26

---

### interactive

---

The `interactive` command loads an interactive REPL shell allowing you to execute repeated commands in real-time against targeted servers.

The following AsciiCast demonstrates its usage.

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/nWylhLFq2SVulV8eU5ZKXAq8e
```

---

See the `Bcome` documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



## CHAPTER 27

---

### tunnel

---

The `tunnel` command forwards a remote port locally.

The following AsciiCast demonstrates how to access a remote elastic search process running on port 9200, by forwarding the connection to local port 9200. In the example, the remote elastic search server is accessible via two intermediate proxies, with Bcome handling all the SSH connectivity transparently.

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/F94gN1eVHYhsitYhDkdQHGxLW
```

---

See the Bcome documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



---

### ping

---

The `ping` command is used to determine whether your SSH configuration functions.

Like most Bcome commands, it can be invoked against individual servers, or against entire namespaces.

The following AsciiCast demonstrates the usage of this command:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/hoa65P4RVyGowcWhwe5mP48QG
```

---

See the Bcome documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



## CHAPTER 29

---

### rsync

---

The `rsync` command uploads a file (or directory, recursively) to all servers in a selection.

---

**Note:** This command name will be altered in a future release to better express its singular function.

---

The following AsciiCast demonstrates the usage of this command:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/pn7d4bJtidsseC9Q3xJHbP3Rg
```

---

See the `Bcome` documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).





## CHAPTER 30

---

### pseudo\_tty

---

The `pseudo_tty` command allows you to execute a command against a server for which you need an interactive shell.

For example, this could be to tail a log, or to access a remote console, such as a MySQL console.

The following AsciiCast illustrates how to utilise pseudo-tty to execute a 'top' command.

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/bYvI2k5i5gXC7oLP9odrmfoZh
```

---

See the Bcome documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



## CHAPTER 31

---

### put\_str

---

The `put_str` command creates a file from a string.

The following AsciiCast demonstrates the usage of this command:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/O2hAKKKXxFyHPsUCeo3uLL2JS
```

---

See the `Bcome` documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



## CHAPTER 32

---

### get

---

The `get` command downloads a file (or directory, recursively) from a remote server.

The following AsciiCast demonstrates the usage of this command:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/EUFijpKypsl4JjvNEWKMN96sx
```

---

See the `Bcome` documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



---

### registry

---

The `registry` command outputs a menu containing your custom registry tasks. You can invoke it at any namespace - in console or terminal mode - to view the registry commands configured for that namespace.

---

**Note:** See our documentation site for [registry configuration](#).

---

The following AsciiCast demonstrates the usage of this command within a Bcome installation containing multiple namespaces, with various registry commands configured within each.

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/6J682AqqD0Wa5tQKlF3M0P8za
```

---

See the Bcome documentation for more information on [executing commands](#).

For a full command list, see [command menu](#).



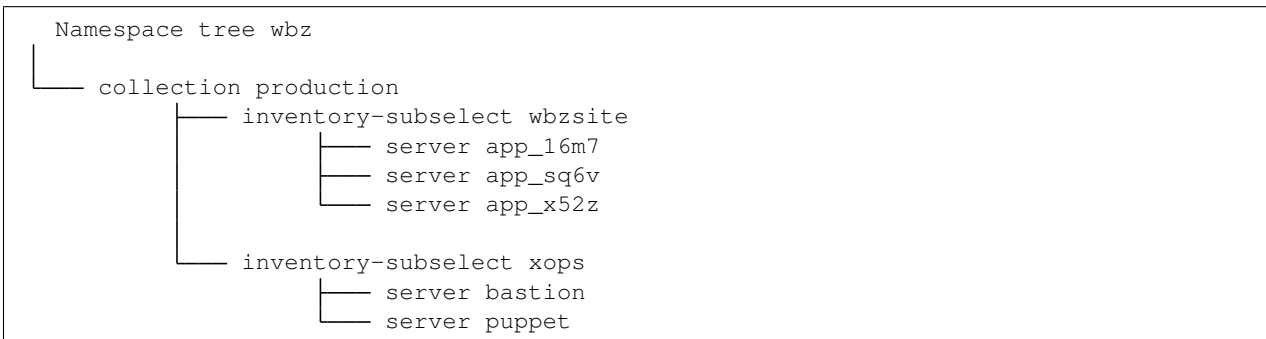


## Shortcuts

A `shortcut` is a command saved in Bcome's registry that can be invoked against any registered namespace using an alternative, or "shortcut" reference.

These are useful either as a shorthand for oft-used commands, or to highlight functionality as part of your Bcome installation.

This guide will use the following Bcome namespace tree:



In our `registry.yml` configuration file we'll associate two commands with the 'production:wbzsite' namespace (an inventory), and one with the `xops:puppet` namespace (a server), as follows:

```

---
"production:wbzsite(.+)?":
- type: shortcut
  description: Http status
  console_command: http_status
  shortcut_command: curl -fI http://webzakimbo.com/
  group: web
- type: shortcut
  description: Thin webserver status
  console_command: thin_status
  shortcut_command: sudo supervisorctl status thin_public
  group: web

```

(continues on next page)

(continued from previous page)

```
"production:xops:puppet":  
- type: shortcut  
  description: Run a 'top' interactively  
  console_command: top  
  shortcut_command: top  
  run_as_pseudo_tty: true  
  group: misc
```

Notice how a regular expression is used to associate the shortcuts. This allows the first two commands to be made available both at inventory & server-level within 'production:wbzsite', and specifically at server-level for our last shortcut.

Note also how our last shortcut is configured as a pseudo-tty. This feature allows shortcuts to access interactive sessions:

---

**Hint:** Use a pseudo-tty interactive shortcut to enable shortcuts to remote command line interfaces, e.g. a MySQL prompt.

---

---

**Note:** For full details on configuring shortcuts, please refer to our documentation site: [shortcuts](#).

---

The following AsciiCast demonstrates the configuration within this guide:

## 34.1 Ascii Cast

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/6SWOI6MMWoyeZya4ttJ17m6QM
```

---

---

## External methods

---

An external-hook registry method allows for invoking an external script, which itself makes use of Bcome's runtime. The namespace context at which the external hook is called is passed to the script.

This guide demonstrates the configuration of a simple external hook.

Here's our script -

```
require 'bcome'
require 'pry'

# Define an orchestrator
orchestrator = ::Bcome::Orchestrator.instance

# Load in the namespace
@node = orchestrator.get(ENV["bcome_context"])

# Work with the namespace

# run a command
command = "echo \"hello world I am `hostname -A`\""
@node.run command

exit 0
```

---

**Hint:** There is a lot you can do with a @node object. See [interacting with node](#).

---

Let's save our script as 'say\_hello\_server.rb' at the following location:

```
.
├── bcome
│   └── scripts
│       └── say_hello_server.rb
```

Next we'll add a registry.yml association for our script, associating it with the 'production:wbzsite' namespace in our namespace tree:

```
---
"production:wbzsite(.+)?":
  - type: external
    description: "Say hello, server"
    console_command: say_hello
    group: salutations
    local_command: ruby scripts/say_hello_server.rb
```

Now let's try out our external registry hook:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/LW2fZbitYWKyaFnCuWtgyJOIP
```

---

**Note:** See our documentation site for more details on configuring [external hooks](#).

---

## CHAPTER 36

---

### Internal Methods

---

Internal registry hooks associate [internal scripts](#) with your Bcome namespaces - they allow for the execution of your internal scripts in the context of the Bcome namespace at which you call them.

This guide demonstrates the configuration of a simple Internal registry hook.

Here's our Internal script class:

```
module ::Bcome::Orchestration
  class MyInternalScript < Bcome::Orchestration::Base

    def execute
      do_something
    end

    def do_something
      # @node is made available to all internal scripts. It is an object representing
      ↪the namespace at which the script was called.
      @node.run command
    end

    def command
      "echo \"Hello, I am `hostname -A`\""
    end
  end
end
```

---

**Hint:** There is a lot you can do with a @node object. See [interacting with node](#).

---

Now let's save our class within our orchestration directory so that it can be loaded into our installation at runtime:

```
└─ bcome
```

(continues on next page)

(continued from previous page)

```
└─ orchestration
   └─ my_internal_script.rb
```

Next we'll add a registry.yml association for our script, associating it with the 'production:wbzsite' namespace in our namespace tree:

```
---
"production:wbzsite(.+)?":
  - type: internal
    description: Say hello, server
    console_command: say_hello
    group: salutations
    orch_klass: MyInternalScript
```

Now let's try out our internal registry hook:

---

**Note:** To replay this AsciiCast in your own terminal, install the `asciinema` package from <https://asciinema.org/>, and then enter the following in your terminal:

```
asciinema play https://asciinema.org/a/5WjMvBMYHJ9VMUjkxlvxZX9EG
```

---

**Note:** See our documentation site for more details on configuring [internal scripts](#).

---