Bossman

Anthony Hogg

Apr 12, 2021

CONTENTS:

1 Install 3

2 Usage 5

3 Quickstarts 9

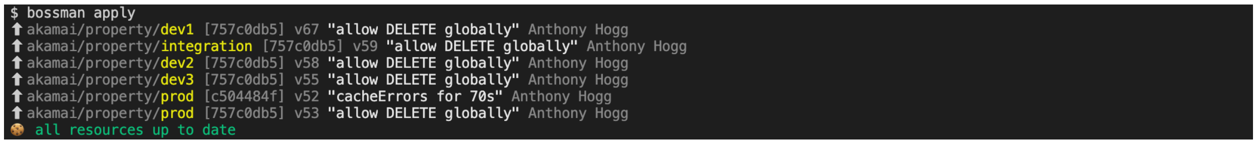
4 Plugins 19

5 Philosophy 23

Bossman is a command line tool for Akamai GitOps. It helps automate the deployment and release management of Akamai product configurations that are maintained in git.

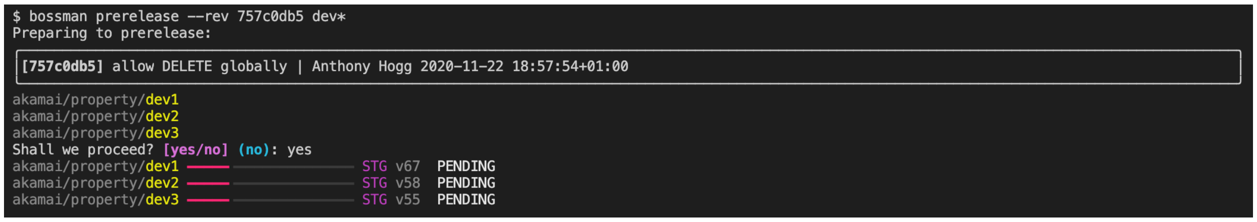
It tries hard to be a very thin glue between git and your infrastructure, providing a simple command line interface to manage the lifecycle of your configuration changes through simple commands:

- `bossman apply` - synchronize git commits to the infrastructure



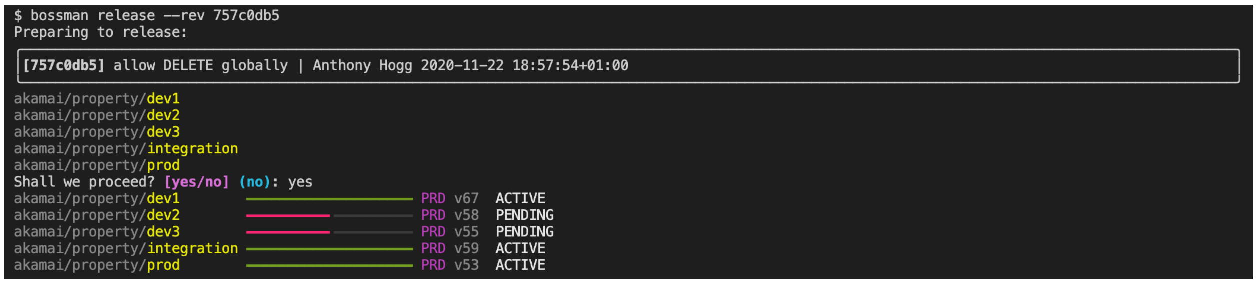
```
$ bossman apply
⬆ akamai/property/dev1 [757c0db5] v67 "allow DELETE globally" Anthony Hogg
⬆ akamai/property/integration [757c0db5] v59 "allow DELETE globally" Anthony Hogg
⬆ akamai/property/dev2 [757c0db5] v58 "allow DELETE globally" Anthony Hogg
⬆ akamai/property/dev3 [757c0db5] v55 "allow DELETE globally" Anthony Hogg
⬆ akamai/property/prod [c504484f] v52 "cacheErrors for 70s" Anthony Hogg
⬆ akamai/property/prod [757c0db5] v53 "allow DELETE globally" Anthony Hogg
🍪 all resources up to date
```

- `bossman prerelease --rev 757c0db5 dev*` - make a specific revision live for testing



```
$ bossman prerelease --rev 757c0db5 dev*
Preparing to prerelease:
 ┌──────────────────────────────────────────────────────────────────────┐
 │[757c0db5] allow DELETE globally | Anthony Hogg 2020-11-22 18:57:54+01:00 │
 └──────────────────────────────────────────────────────────────────────┘
akamai/property/dev1
akamai/property/dev2
akamai/property/dev3
Shall we proceed? [yes/no] (no): yes
akamai/property/dev1 ━━━━━────────────── STG v67  PENDING
akamai/property/dev2 ━━━━━────────────── STG v58  PENDING
akamai/property/dev3 ━━━━━────────────── STG v55  PENDING
```

- `bossman release --rev 757c0db5` - make a specific revision live to end users

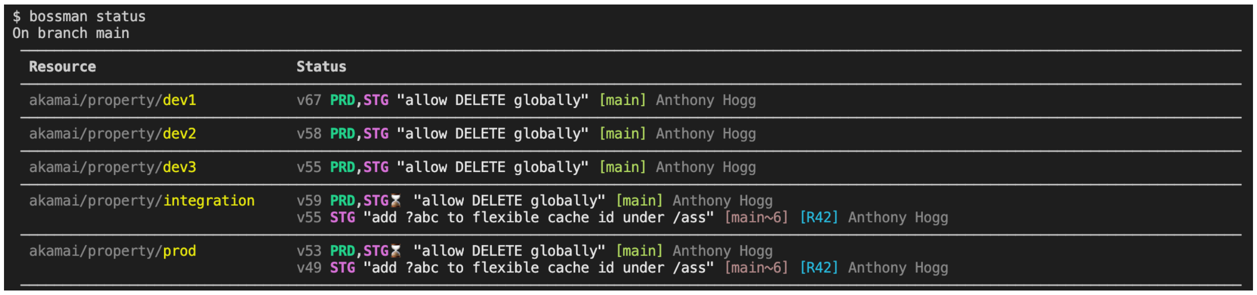


```
$ bossman release --rev 757c0db5
Preparing to release:
 ┌──────────────────────────────────────────────────────────────────────┐
 │[757c0db5] allow DELETE globally | Anthony Hogg 2020-11-22 18:57:54+01:00 │
 └──────────────────────────────────────────────────────────────────────┘
akamai/property/dev1
akamai/property/dev2
akamai/property/dev3
akamai/property/integration
akamai/property/prod
Shall we proceed? [yes/no] (no): yes
akamai/property/dev1        ━━━━━━━━━━━━━━━━━━━━ PRD v67  ACTIVE
akamai/property/dev2        ━━━━━━━━━━────────── PRD v58  PENDING
akamai/property/dev3        ━━━━━━━━━━────────── PRD v55  PENDING
akamai/property/integration ━━━━━━━━━━━━━━━━━━━━ PRD v59  ACTIVE
akamai/property/prod        ━━━━━━━━━━━━━━━━━━━━ PRD v53  ACTIVE
```

- `bossman status` - show the state of the infrastructure

Because Bossman is simply built "on top of git", it is possible to implement many workflows with ease.

- One single environment
- Linear promotional deployment - dev, integration, qa, preprod, prod
- Distributed development - work in parallel on multiple dev environments
- SaaS or regional model - one template and many production environments with small variations



```
$ bossman status
On branch main
 ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
  Resource                        Status
 ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
  akamai/property/dev1            v67 PRD,STG "allow DELETE globally" [main] Anthony Hogg
 ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
  akamai/property/dev2            v58 PRD,STG "allow DELETE globally" [main] Anthony Hogg
 ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
  akamai/property/dev3            v55 PRD,STG "allow DELETE globally" [main] Anthony Hogg
 ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
  akamai/property/integration     v59 PRD,STG⏳ "allow DELETE globally" [main] Anthony Hogg
                                  v55 STG "add ?abc to flexible cache id under /ass" [main~6] [R42] Anthony Hogg
 ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
  akamai/property/prod            v53 PRD,STG⏳ "allow DELETE globally" [main] Anthony Hogg
                                  v49 STG "add ?abc to flexible cache id under /ass" [main~6] [R42] Anthony Hogg
 ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

CHAPTER

ONE

INSTALL

1.1 Requirements

Bossman requires Python >=3.8 and git >=2.3.10.

Windows

Bossman should just about run on Windows, but support as it is will be limited. Extra dependencies may be required (pay close attention to the output on the console when installing via pip).

It is highly recommended to use either the Windows Subsystem for Linux or Docker when running bossman on windows.

The Windows Subsystem for Windows will provide a much smoother than native, but do pay close attention to the warnings on the console.

In both cases (native or WSL), it is strongly recommended to install the Windows Terminal. The default terminal font does not support all the glyphs used by bossman to convey resource status information.

1.2 Installing Locally

```
python3 -m pip install bossman
```

1.3 Using Docker

Bossman provides public docker images which can be convenient if the above requirements are hard to meet.

```
docker run -it --rm \
   -e TERM=xterm-256color \
   -v ~/.ssh/id_rsa:/home/bossman/.ssh/id_rsa:ro \
   -v $PWD:/work \
   -v ~/.edgerc:/home/bossman/.edgerc ynohat/bossman version
```

- `-i` is required to keep standard input open, which is required for displaying progress bars in the terminal
- `-t` is required to allocate a pseudo-terminal
- `-e TERM=xterm-256color` gets rich colour output, which is necessary in particular for making the best use of `bossman status`

- `-v ~/.ssh/id_rsa:/home/bossman/.ssh/id_rsa` mounts your SSH private key into the container; see notes below
- `-v $PWD:/work` mounts the current working directory to `/work` which is the working directory in the container
- `-v ~/.edgerc:/home/bossman/.edgerc` mounts the Akamai credential file in the appropriate location for bossman to find them
- `ynohat/bossman` references the Docker repository, however you may wish to target a specific tag.

It is recommended to create a shell alias to avoid typing the above repeatedly!

about mounting your SSH key

Bossman needs to interact with the git remotes, over SSH. Ideally, we would forward the SSH agent socket to docker, but I am failing to get this to work on Mac (and maybe Windows as well). There seems to be an issue with SSH agent socket forwarding when docker is running inside a hypervisor.

As a result, mounting the private key is the only way I can propose for now, along with assurances that bossman will not attempt to steal these credentials.

I cannot however extend the same guarantee to bossman's dependencies.

If you choose to go forward with mounting your private key as suggested, please make sure you mount the key that you have setup in the git remote, of course.

Important when using docker, the git `user.name` and `user.email` are not set globally. Make sure you set them locally in the repository configuration:

```
git config --local user.name "Jane DOE"
git config --local user.email "Jane.DOE@acme.org"
```

The docker image also comes with a few other tools that go well with bossman, in particular:

- `jq` and `git`
- `jsonnet` and `jsonnetfmt` in support of the Jsonnet templating language
- The `akamai` command, along with the `akamai jsonnet` plugin which makes it easier to work with Akamai configuration as Jsonnet
- `httpie` and the `httpie-edgegrid` authentication plugin

CHAPTER

TWO

USAGE

Bossman works on top of git working copies. Within a git working copy, Bossman will manage resources using plugins according to a simple configuration file.

This page describes how to configure and operate bossman.

2.1 Configuration

Configuration tells bossman where resources are in the repository, and which plugin to use to manage them.

It lives in a YAML file which is optional if the defaults are acceptable.

It should be called `.bossman` and reside at the root of the repository. It should be added to source control.

It should have a single `resources` field defining associations between file match patterns and resource plugins:

```
resources:
- module: bossman.plugins.akamai.property
  pattern: akamai/property/{name}
```

The `module` field must be an importable module on the python path. Multiple resources may reference the same module.

The `pattern` field is a path, relative to the root of the repository. `{placeholders}` like `{name}` in the example above will always match a single path component (no slashes). The supported placeholders are specific to each plugin.

In addition to the `module` and `pattern` fields, each resouce group can define additional plugin-specific parameters in an `options` field.

See the *Plugins* documentation pages for details.

.gitignore

Bossman creates a `.bossmancache` file at the root of the repository containing cache entries to speed up specific lookups. This should be added to `.gitignore`.

2.2 Targeting resources

The `glob` argument is accepted by all bossman commands that interact with resources. It allows the operator to restrict the command to a subset of resources. For example, to get the status of all dev resources:

```
bossman status dev*
```

It can be provided multiple times, which will restrict operation to the subset of resources whose paths match any of the patterns.

Path matching details

It performs partial matching on resource paths using Unix filename pattern matching.

Table 1: Pattern modifiers

Pattern	Meaning
`*`	Matches everything (including /)
`?`	Matches any single character (including /)
`[seq]`	Matches any character in _seq_
`[!seq]`	Matches any character _not_ in _seq_

Assuming you have the following resources in your repository:

akamai/property/dev1
akamai/requestcontrol/dev1
akamai/property/dev2
akamai/requestcontrol/dev2
akamai/property/dev3
akamai/requestcontrol/dev3
akamai/property/integration
akamai/property/prod

- `akamai`, `akam` or `akamai/*`: will select all the resources
- `property` or `akamai/property`: will select all Akamai properties
- `dev[1-2]` will select all Akamai resources (properties and requestcontrol) for dev1 and dev2
- `dev[!3]` will select all Akamai resources (properties and requestcontrol) for dev1 and dev2

Combining with shell expansion

Some shells, such as `bash` and `zsh` also support expansion patterns that can complement bossman's pattern matching for very convenient operation. For example, to select all non-production resources with the set of resources above:

```
bossman status property/{dev\*,integration}
# gets expanded to the following by the shell
# bossman status property/dev* property/integration
```

2.3 bossman version

This command outputs the version. It is the only command that can be run before `bossman init`.

2.4 bossman init

This command must be run before anything can be done by Bossman. It adjusts the `.git/config` file, adds a `[bossman]` section and extra refspecs to all remotess, to ensure that git notes are properly pushed and pulled along with commits.

2.5 bossman status [glob*]

Provides synthetic information about the state of resources managed by bossman.

2.6 bossman apply [glob*]

Deploys all pending commits.

2.7 bossman validate [glob*]

Validates the correctness of resources in the working copy.

This is the only command that does not operate on a commit.

2.8 bossman (pre)prerelease [--rev HEAD] [glob*]

- `prerelease`: makes a given revision available to an internal audience, typically for testing
- `release`: makes a given revision available to the end users

`--rev` can be any valid git commit reference, e.g.

- a commit hash (full or abbreviated)
- a tag name
- a branch name
- `HEAD`
- a relative ref

2.9 `bossman log [glob*]`

Outputs the revision history of the selected resources.

CHAPTER

THREE

QUICKSTARTS

These short tutorials guide you through simple tasks with Bossman.

First ensure that you can successfully run the `bossman` command after following the installation instructions (*Install*).

3.1 Akamai : Simple Property

In this scenario, we will create a CDN property from scratch and make a simple change.

3.1.1 Requirements

- Bossman (*Install*)
- Access to Akamai Control Center
- (Recommended) A dedicated Access Control Group (ACG, a folder in the control center)
- (Recommended) Akamai OPEN credentials restricted to the ACG with read-write PAPI v1 privileges
- A Content Provider Code
- A domain name (this tutorial will refer to `example.com`)
- A Standard TLS Akamai Edge Hostname
- A `$HOME/.edgerc` file with a `papi` section (but you can configure that after reading *Akamai Property*)

We will be using https://httpbin.org as an origin server, but you are free to use a different one if you prefer.

3.1.2 Initializing the repository

```
git init bossman_quickstart_akamai_simple_property
```

Now that you have your git repository, Bossman needs a quick initialization as well.

```
cd bossman_quickstart_akamai_simple_property
bossman init
```

All set.

3.1.3 Preparing the directory structure

By default, bossman expects the following tree structure:

```
$ tree
.
└── akamai
    └── property
        └── bossman_from_scratch
            ├── hostnames.json
            └── rules.json

3 directories, 2 files
```

The `akamai/property` prefix identifies the plugin to use. The Akamai property plugin expects subdirectories to be property names. With this file tree, we will be managing a property called `bossman_from_scratch`.

Two Akamai properties cannot have the same name within an account!

Creating this structure is trivial:

```
mkdir -p akamai/property/bossman_from_scratch
```

3.1.4 Creating the `hostnames.json` file

This file describes the hostname mapping for the property, more information here:

PAPI V1 : PUT Property Hostnames.

You can use this as a template

```
[
  {
      "cnameFrom": "bossmanfromscratch.example.com",
      "cnameTo": "your-edge-hostname.edgesuite.net",
      "cnameType": "EDGE_HOSTNAME"
  }
]
```

Simply make sure you replace `example.com` and `your-edge-hostname` with your own values.

3.1.5 Creating the `rules.json` file

This file describes the delivery and caching rules that should be applied to traffic served by the configuration.

It obeys a schema described in the PAPI Feature Catalog Reference. This schema is versioned, and we will use a frozen version for the sake of stability in this tutorial.

You can use the following as a template:

```
{
  "contractId": "YOUR_CONTRACT_ID",
  "groupId": "YOUR_GROUP_ID",
  "productId": "YOUR_PRODUCT_ID",
  "ruleFormat": "v2020-03-04",
  "rules": {
      "name": "default",
```

(continues on next page)

(continued from previous page)

```
      "comments": "The behaviors in the Default Rule apply to all requests.",
      "options": {
        "is_secure": false
      },
      "behaviors": [
        {
            "name": "origin",
            "options": {
              "cacheKeyHostname": "ORIGIN_HOSTNAME",
              "compress": true,
              "customValidCnValues": [
                  "{{Origin Hostname}}",
                  "{{Forward Host Header}}"
              ],
              "enableTrueClientIp": false,
              "forwardHostHeader": "REQUEST_HOST_HEADER",
              "hostname": "httpbin.org",
              "httpPort": 80,
              "httpsPort": 443,
              "originCertsToHonor": "STANDARD_CERTIFICATE_AUTHORITIES",
              "originSni": true,
              "originType": "CUSTOMER",
              "standardCertificateAuthorities": [
                  "akamai-permissive"
              ],
              "verificationMode": "PLATFORM_SETTINGS"
            }
        },
        {
            "name": "cpCode",
            "options": {
              "value": {
                  "id": YOUR_CPCODE_ID
              }
            }
        },
        {
            "name": "caching",
            "options": {
              "behavior": "MAX_AGE",
              "mustRevalidate": false,
              "ttl": "31d"
            }
        }
      ]
  }
}
```

3.1.6 Quick validation

Bossman can help you validate your working copy using this command:

```
bossman validate
```

This will list each resource with a thumbs up emoji if validation passed.

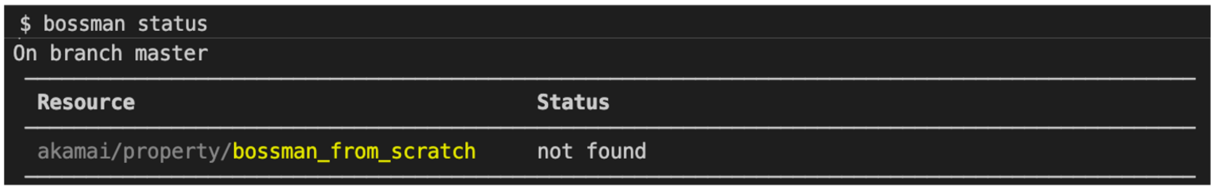
This is only a very superficial validation that helps with:
- JSON syntax errors
- validity according to the schema

3.1.7 First commit & deployment

Bossman does not deploy from working copy, so we need to commit our changes.

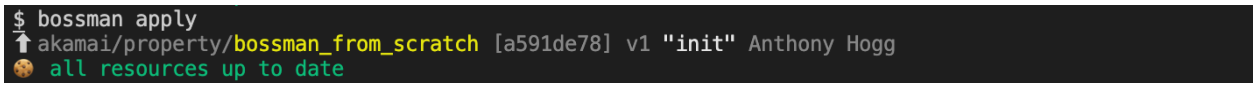
```
git add akamai
git commit -m "init"
```

Before deploying this change, we can check the status:



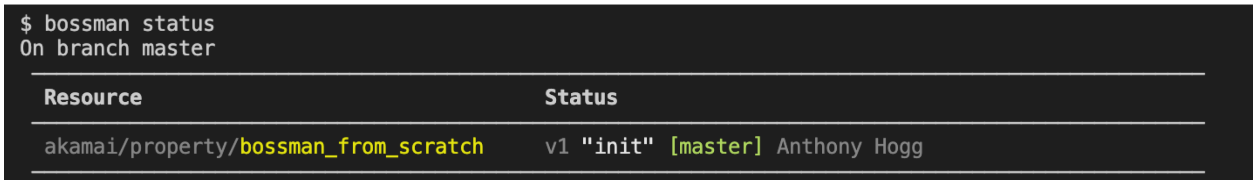
```
$ bossman status
On branch master
 ─────────────────────────────────────────────────────────────────────
  Resource                                Status
 ─────────────────────────────────────────────────────────────────────
  akamai/property/bossman_from_scratch    not found
 ─────────────────────────────────────────────────────────────────────
```

We can create the property and deploy the change from here:



```
$ bossman apply
⬆ akamai/property/bossman_from_scratch [a591de78] v1 "init" Anthony Hogg
🍪 all resources up to date
```

And we can look at the status again:



```
$ bossman status
On branch master
 ─────────────────────────────────────────────────────────────────────
  Resource                                Status
 ─────────────────────────────────────────────────────────────────────
  akamai/property/bossman_from_scratch    v1 "init" [master] Anthony Hogg
 ─────────────────────────────────────────────────────────────────────
```

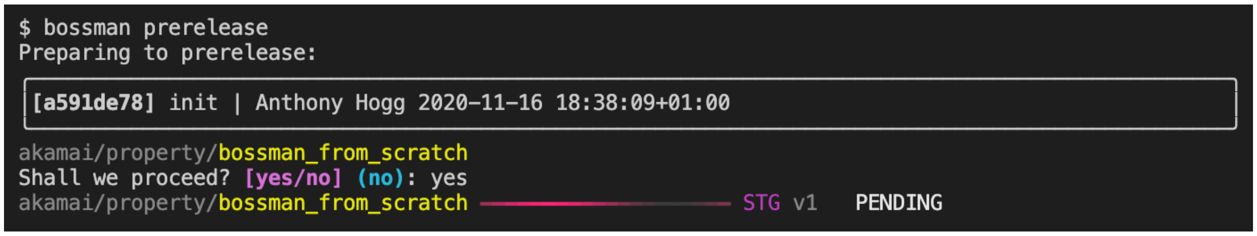
3.1.8 Activating to staging

Now we are ready to activate the property to staging.

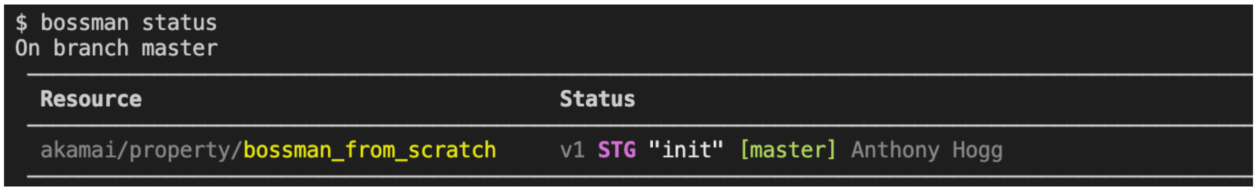
After the process completes, we can check the status again:

We have an indication that v1 is active on the staging network.

And we can see the result of our efforts in Akamai Control Center!



```
$ bossman prerelease
Preparing to prerelease:
 [a591de78] init | Anthony Hogg 2020-11-16 18:38:09+01:00

akamai/property/bossman_from_scratch
Shall we proceed? [yes/no] (no): yes
akamai/property/bossman_from_scratch ━━━━━━━━━━━━━━━━━━━━ STG v1   PENDING
```



```
$ bossman status
On branch master
  Resource                                  Status
  akamai/property/bossman_from_scratch      v1 STG "init" [master] Anthony Hogg
```

3.1.9 Congratulations!

Next step: *Akamai : Parallel Environments*

3.2 Akamai : Parallel Environments

In this scenario, we will build on the *Akamai : Simple Property* quickstart. and add a preproduction environment.

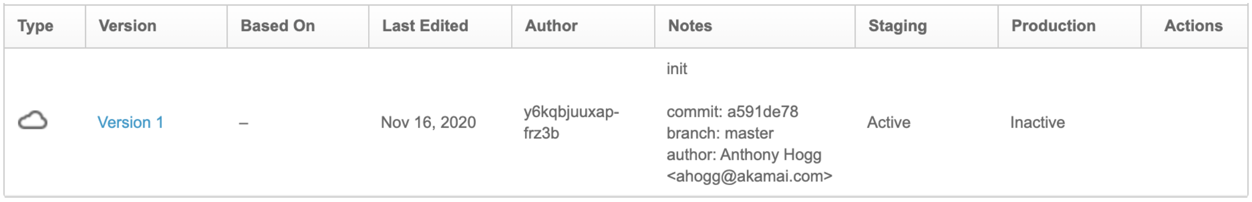
3.2.1 Requirements

- You have completed the *Akamai : Simple Property* quickstart

3.2.2 Where we are

You should have the following in your repository:

```
$ tree
.
└── akamai
    └── property
        └── bossman_from_scratch
            ├── hostnames.json
            └── rules.json

3 directories, 2 files
```



Type	Version	Based On	Last Edited	Author	Notes	Staging	Production	Actions
☁	Version 1	–	Nov 16, 2020	y6kqbjuuxap-frz3b	init commit: a591de78 branch: master author: Anthony Hogg <ahogg@akamai.com>	Active	Inactive	

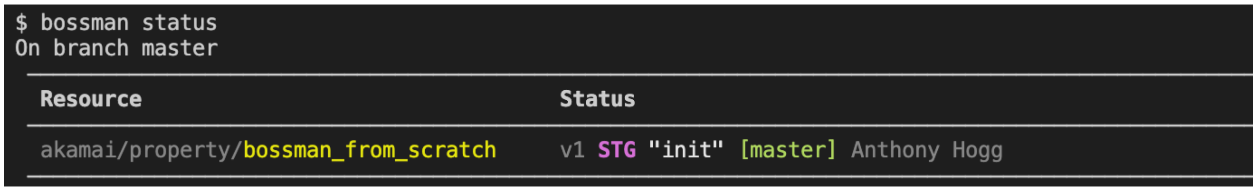
3.2.3 Adding preproduction

```
cp -R akamai/property/bossman_from_scratch{,_preprod}
```

Important: We must now update the `hostnames.json` file so that it serves a different hostname.

```
[
  {
      "cnameFrom": "bossmanfromscratch-preprod.example.com",
      "cnameTo": "your-edge-hostname.edgesuite.net",
      "cnameType": "EDGE_HOSTNAME"
  }
]
```

Let's make a quick detour now and run `bossman status`:

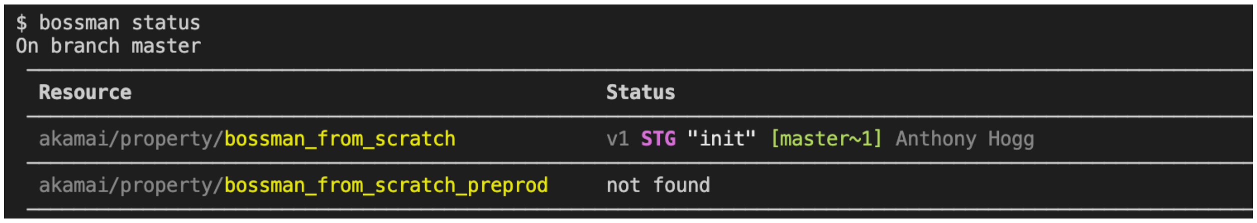


```
$ bossman status
On branch master

  Resource                                  Status

  akamai/property/bossman_from_scratch      v1 STG "init" [master] Anthony Hogg
```

Our new environment is not shown! This is because bossman only concerns itself with commits, by design.

```
git add akamai/property/bossman_from_scratch_preprod
git commit -m "add preproduction"
```

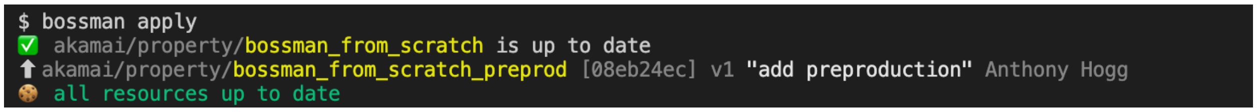
Now the new environment is tracked by bossman:



```
$ bossman status
On branch master

  Resource                                          Status

  akamai/property/bossman_from_scratch              v1 STG "init" [master~1] Anthony Hogg

  akamai/property/bossman_from_scratch_preprod      not found
```

3.2.4 Deployment

Just as we did previously, it is now time to deploy our change:

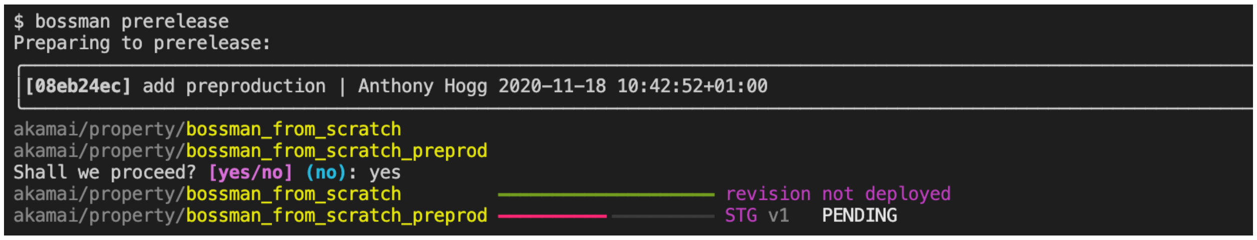


```
$ bossman apply
✅ akamai/property/bossman_from_scratch is up to date
⬆akamai/property/bossman_from_scratch_preprod [08eb24ec] v1 "add preproduction" Anthony Hogg
🍪 all resources up to date
```

There was no change to deploy to the `bossman_from_scratch` property, and v1 of `bossman_from_scratch_preprod` was created as expected.

3.2.5 Prerelease

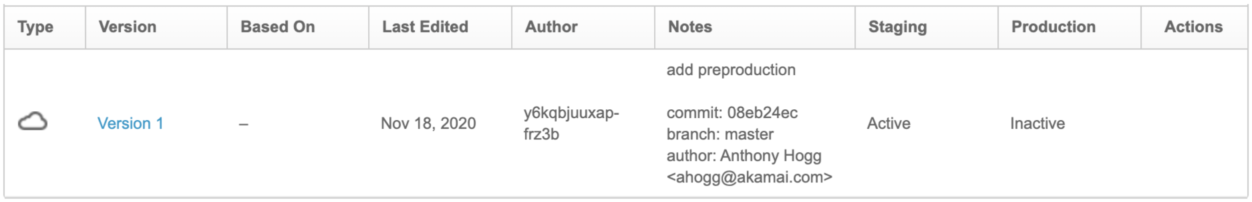
Now let us prerelease our new configuration to the staging network.



```
$ bossman prerelease
Preparing to prerelease:
 [08eb24ec] add preproduction | Anthony Hogg 2020-11-18 10:42:52+01:00
akamai/property/bossman_from_scratch
akamai/property/bossman_from_scratch_preprod
Shall we proceed? [yes/no] (no): yes
akamai/property/bossman_from_scratch         ━━━━━━━━━━━━━━━━━━━━ revision not deployed
akamai/property/bossman_from_scratch_preprod ━━━━━━━━━━━━━━━━━━━━ STG v1    PENDING
```

Note that because the commit did not touch `bosssman_from_scratch`, it was not deployed, so bossman does not attempt to activate it.

We can check that all is in order in Akamai Control Center:



Type	Version	Based On	Last Edited	Author	Notes	Staging	Production	Actions
☁	Version 1	–	Nov 18, 2020	y6kqbjuuxap-frz3b	add preproduction commit: 08eb24ec branch: master author: Anthony Hogg <ahogg@akamai.com>	Active	Inactive	

3.2.6 Making a change on both environments

By default, only HTTP `GET` requests are allowed on the Akamai platform. Allowing the use of more methods is quite easy, though. Let's support `POST`!

Simply add lines 44-51 highlighted below to both files:

- akamai/property/bossman_from_scratch/rules.jsonn
- akamai/property/bossman_from_scratch_preprod/rules.jsonn

Do NOT copy paste the entire JSON here, since it contains placeholders that you already filled in with different values previously.

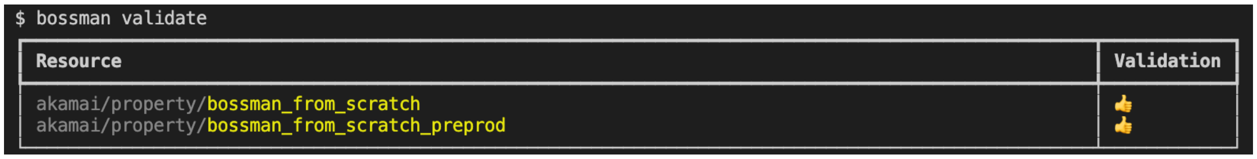
```
1  {
2    "contractId": "YOUR_CONTRACT_ID",
3    "groupId": "YOUR_GROUP_ID",
4    "productId": "YOUR_PRODUCT_ID",
5    "ruleFormat": "v2020-03-04",
6    "rules": {
7        "name": "default",
8        "comments": "The behaviors in the Default Rule apply to all requests.",
9        "options": {
10         "is_secure": false
11       },
12       "behaviors": [
13         {
14             "name": "origin",
15             "options": {
16               "cacheKeyHostname": "ORIGIN_HOSTNAME",
17               "compress": true,
18               "customValidCnValues": [
```

(continues on next page)

(continued from previous page)

```
                  "{{Origin Hostname}}",
                  "{{Forward Host Header}}"
              ],
              "enableTrueClientIp": false,
              "forwardHostHeader": "REQUEST_HOST_HEADER",
              "hostname": "httpbin.org",
              "httpPort": 80,
              "httpsPort": 443,
              "originCertsToHonor": "STANDARD_CERTIFICATE_AUTHORITIES",
              "originSni": true,
              "originType": "CUSTOMER",
              "standardCertificateAuthorities": [
                  "akamai-permissive"
              ],
              "verificationMode": "PLATFORM_SETTINGS"
            }
        },
        {
            "name": "cpCode",
            "options": {
              "value": {
                  "id": YOUR_CPCODE_ID
              }
            }
        },
        {
            "name": "caching",
            "options": {
              "behavior": "MAX_AGE",
              "mustRevalidate": false,
              "ttl": "31d"
            }
        },
        {
          "name": "allowPost",
          "options": {
             "allowWithoutContentLength": false,
             "enabled": true
          }
       }
    ]
  }
}
```

Now use `bossman validate` to run superficial syntax checks on your working copy…



```
$ bossman validate
┌──────────────────────────────────────────────────┬────────────┐
│ Resource                                         │ Validation │
├──────────────────────────────────────────────────┼────────────┤
│ akamai/property/bossman_from_scratch             │ 👍         │
│ akamai/property/bossman_from_scratch_preprod     │ 👍         │
└──────────────────────────────────────────────────┴────────────┘
```

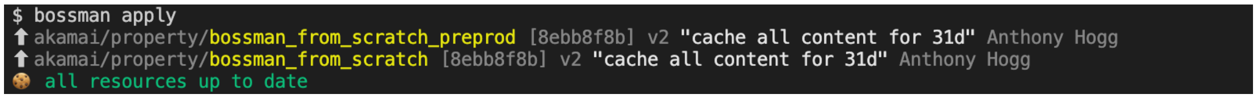
If you get thumbs up, great! If not, double-check the JSON.

We can now commit the change:

```
git commit -am "allow POST"
```

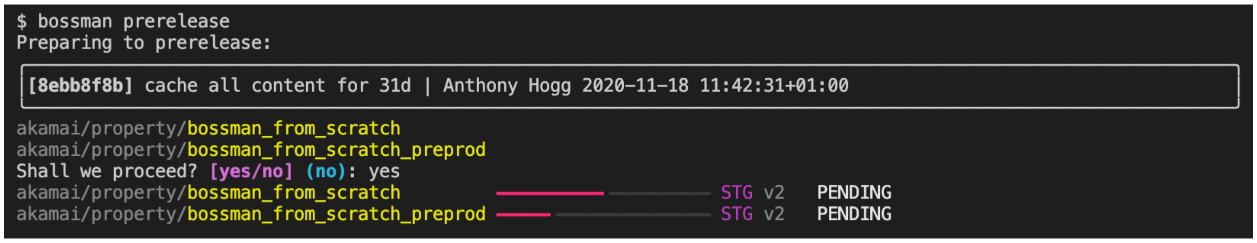
. . . and deploy the change - you're getting used to this by now :)

```
Note that the commit message displayed in the screenshot is from an older version of␣
↪this
tutorial where we were effecting a different change, please bear with this.
```



```
$ bossman apply
⬆ akamai/property/bossman_from_scratch_preprod [8ebb8f8b] v2 "cache all content for 31d" Anthony Hogg
⬆ akamai/property/bossman_from_scratch [8ebb8f8b] v2 "cache all content for 31d" Anthony Hogg
🍪 all resources up to date
```

Now, we can activate on the Akamai staging network:



```
$ bossman prerelease
Preparing to prerelease:

 [8ebb8f8b] cache all content for 31d | Anthony Hogg 2020-11-18 11:42:31+01:00

akamai/property/bossman_from_scratch
akamai/property/bossman_from_scratch_preprod
Shall we proceed? [yes/no] (no): yes
akamai/property/bossman_from_scratch                      STG v2   PENDING
akamai/property/bossman_from_scratch_preprod              STG v2   PENDING
```

3.2.7 Closing Remarks

Bossman made it very easy to deploy and activate the configurations. But a few things could be improved:

- **You repeated yourself** when setting up the `allowPost` behaviour, this is (sometimes) an engineering anti-pattern and there would be value in avoiding it!

Because bossman does not concern itself with how you build the configuration JSON and only cares about how it is changed over time, you can use a template tool as a valuable complement. This will allow you to manage your core configuration template in one place and automatically specialize it for the different environments you maintain.

- **It would have been nice to test the caching behaviour on preprod first** if this is your usual workflow. . .

In this tutorial we showed how all configurations could be kept in lockstep, but bossman supports the "preprod then prod" workflow with equal ease.

We will cover these in later tutorials.)

CHAPTER

FOUR

PLUGINS

4.1 Akamai Property

This page provides reference information about how bossman commands relate to Akamai Property management.

4.1.1 Resource Configuration

```yaml
resources:
  - module: bossman.plugins.akamai.property
    pattern: akamai/property/{name}
    options:
      edgerc: ~/.edgerc
      section: papi
      #switch_key: xyz
```

The above are the default values, applied even if the `.bossman` configuration file is not present. You only need to configure if you need to depart from the defaults.

With these defaults, Bossman will look for folders under `akamai/property` and treat them as Akamai Property configurations. The `{name}` placeholder is required and defines the name of the property to be managed.

The next section details the structure of the resource, the files Bossman expects to find within the property configuration folder.

4.1.2 Resource Structure

An Akamai property is composed of two files:

- `hostnames.json` with the host -> edgehostname mapping information
- `rules.json` with the CDN delivery configuration rule tree to apply

Both files are completely standard as per the documented schemas and can be used independently of Bossman through regular API calls or other automation tools.

Bossman imposes the presence of top-level fields in `rules.json` which are not required by the schema:

- `productId`
- `ruleFormat`
- `groupId`
- `contractId`

`productId` and `ruleFormat` are required so that Bossman can accurately validate and version freeze property versions when they are deployed.

`groupId` and `contractId` are required so that Bossman has enough information to create new properties in the appropriate location.

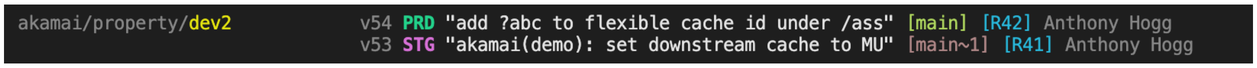
4.1.3 `bossman status [glob*]`

The `status` command displays details about *interesting* property versions.

Interesting property versions are either:

- activating, or pending activation on any network
- the latest version
- deployed versions of any HEAD commit of any active branch

In the normal case, property versions are created by bossman and their status line shows:



```
akamai/property/dev2          v54 PRD "add ?abc to flexible cache id under /ass" [main] [R42] Anthony Hogg
                              v53 STG "akamai(demo): set downstream cache to MU" [main~1] [R41] Anthony Hogg
```

- the property version
- STG, PRD or STG,PRD depending on the activation status (if they are pending activation to staging or production, the network trigram is followed by an hourglass)
- a icon if the version has validation errors
  - that this icon should normally never be visible alongside an activation indicator (STG,PRD)
  - this indicator relies on information stored in git at `apply` time (for performance). This means that a `dirty` version will not show validation errors
- the first line of the property version notes, truncated to 40 characters
- a series of git refs to the corresponding commit, coloured green if the version corresponds to the latest commit on that branch, or brown if it is behind
- a series of tags pointing at the corresponding commit, coloured blue

See *Making changes in the UI* for more details about handling dirty versions.

4.1.4 `bossman apply [--force] [glob*]`

The `apply` command creates a new version for every commit on the current branch.

If the property does not exist, it is created.

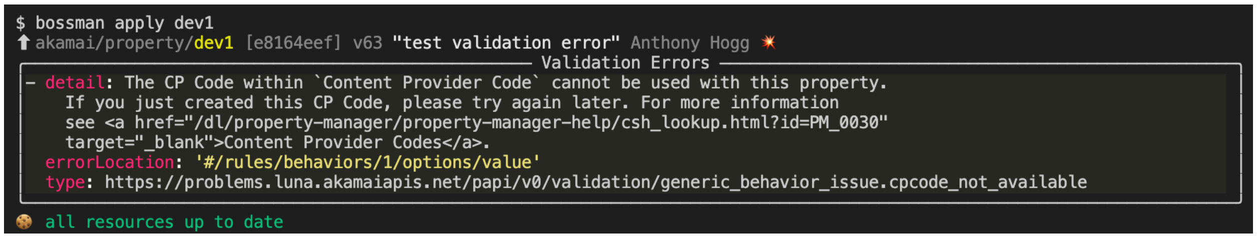
The `productId` and `ruleFormat` fields specified in the `rules.json` file are used to freeze the property version to a specific schema version.

If the property version has validation errors, `apply` will succeed but a icon will be displayed, along with a list of errors as reported by the PAPI endpoint:

If bossman detects that the latest version of the property is `dirty`, it will skip applying unless the `--force` flag is also provided.

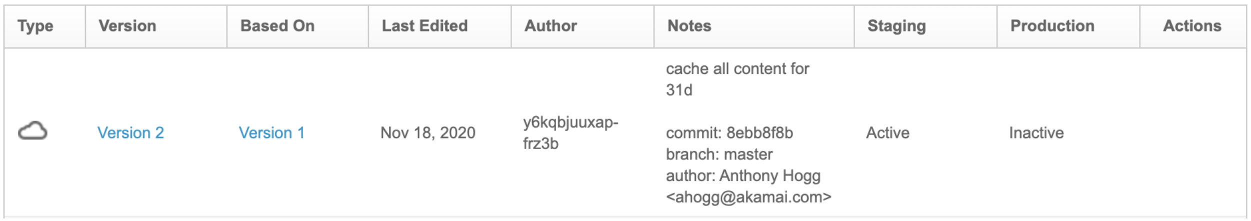
Bossman structures property version notes, by encoding:

- the commit message



```
$ bossman apply dev1
⬆akamai/property/dev1 [e8164eef] v63 "test validation error" Anthony Hogg 💥
┌─────────────────────────────────── Validation Errors ───────────────────────────────────┐
│- detail: The CP Code within `Content Provider Code` cannot be used with this property.   │
│    If you just created this CP Code, please try again later. For more information         │
│    see <a href="/dl/property-manager/property-manager-help/csh_lookup.html?id=PM_0030"    │
│    target="_blank">Content Provider Codes</a>.                                            │
│  errorLocation: '#/rules/behaviors/1/options/value'                                       │
│  type: https://problems.luna.akamaiapis.net/papi/v0/validation/generic_behavior_issue.cpcode_not_available │
└───────────────────────────────────────────────────────────────────────────────────────────┘
🍪 all resources up to date
```

- metadata about the commit, including
  - the abbreviated commit hash
  - the branches containing the commit
  - the author
  - if applicable, the committer



Type	Version	Based On	Last Edited	Author	Notes	Staging	Production	Actions
☁	Version 2	Version 1	Nov 18, 2020	y6kqbjuuxap-frz3b	cache all content for 31d commit: 8ebb8f8b branch: master author: Anthony Hogg <ahogg@akamai.com>	Active	Inactive	

The purpose is threefold.

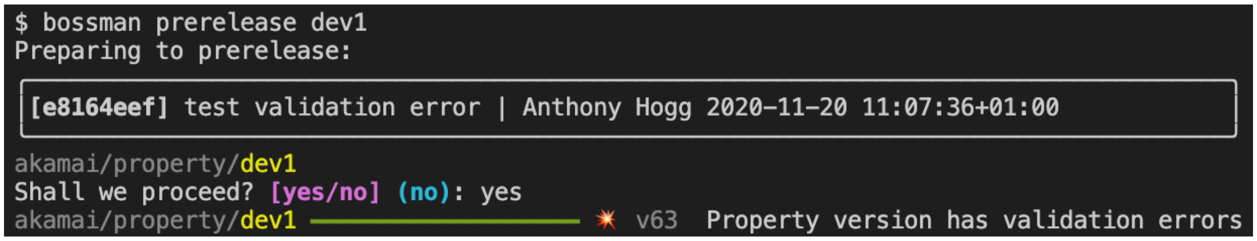
- It improves the quality of property version notes; if a good git commit message convention is in place, it is automatically applied to the property version;
- The author(s) of the change are referenced clearly, which helps because API calls do not record this information in a legible way in the regular Author field;
- It provides a mechanism for bossman to correlate property versions with git revisions

4.1.5 `bossman (pre)release [--rev HEAD] [glob*]`

prerelease : activates the selected revision and resources to the staging network

release : the same, to the production network

If the property version has validation errors, activation is disallowed:



```
$ bossman prerelease dev1
Preparing to prerelease:

 ┌──────────────────────────────────────────────────────────────────────┐
 │[e8164eef] test validation error | Anthony Hogg 2020-11-20 11:07:36+01:00 │
 └──────────────────────────────────────────────────────────────────────┘
akamai/property/dev1
Shall we proceed? [yes/no] (no): yes
akamai/property/dev1 ━━━━━━━━━━━━━━━━━━ 💥 v63  Property version has validation errors
```

Acivation notes & notifications

When Bossman triggers an activation, it automatically adds the following emails to the email notification list:

- the author of the commit being released
- the committer of the commi being released, if different from the auhor
- the currently configured git user

It also formats the activation notes to look like this:

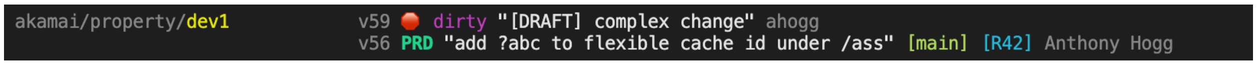
```
activation of 6d4fcb37 (R41) by jane.doe@acme.org using bossman 0.25.0
```

Where

- `6d4fcb37` is the abbreviated commit hash being released
- `R41` is a list of tags pointing at the commit
- `jane.doe@acme.org` is the current git user email

4.1.6 Making changes in the UI

It is entirely acceptable to create new versions in the UI without breaking bossman. If an interesting version was created without using bossman, it will be called out as **dirty**, and will lack any git ref information to relate it to git history :



```
akamai/property/dev1           v59 🔴 dirty "[DRAFT] complex change" ahogg
                               v56 PRD "add ?abc to flexible cache id under /ass" [main] [R42] Anthony Hogg
```

There are two caveats however:

- **bossman will not be able to activate these versions** - indeed, bossman concerns itself with the deployment and release cycle of *git commits*. By definition, a dirty version is not associated to a commit, and is therefore “out of band”; the recommended approach is then to re-integrate the change into the code
- **bossman cannot help with reintegration of changes from dirty versions** - this needs to be done manually and the method will depend largely on how the configurations are maintained as code.

These aspects are by design and unlikely to change. Bossman acknowledges the need to make occasional changes in the UI, but if it is the primary workflow, then maybe bossman is not the best choice.

CHAPTER

FIVE

PHILOSOPHY

When announcing Waypoint, Hashicorp defined the application lifecycle as having three main stages.

- Build : convert application source code to an artifact
- Deploy : push the built artifact to a platform
- Release : make the artifact, on the platform, available to its public

This is a simplified, but useful view.

When managing Akamai configuration artifacts, you are:

- Building JSON configuration from a template
- Deploying JSON configuration to a new version of an Akamai object (property, cloudlet policy etc. . .)
- Releasing to the staging network or the production network

Bossman is not opinionated about the build process. It concerns itself mainly with deployment and release of configuration artifacts that are built through some other mechanism.

About the author

My name is Anthony Hogg, and I am an Enterprise Architect working at Akamai Technologies. I maintain Bossman to help my customers apply DevOps methodologies to their Akamai configurations in those cases where existing tooling is not a perfect fit.

Disclaimer

Bossman is not officially supported by Akamai Technologies. I provide support, fixes and features on a best-effort basis with a priority on the requirements of my customers, but I also try to be as responsive as possible, and welcome contributions.