Kapitan

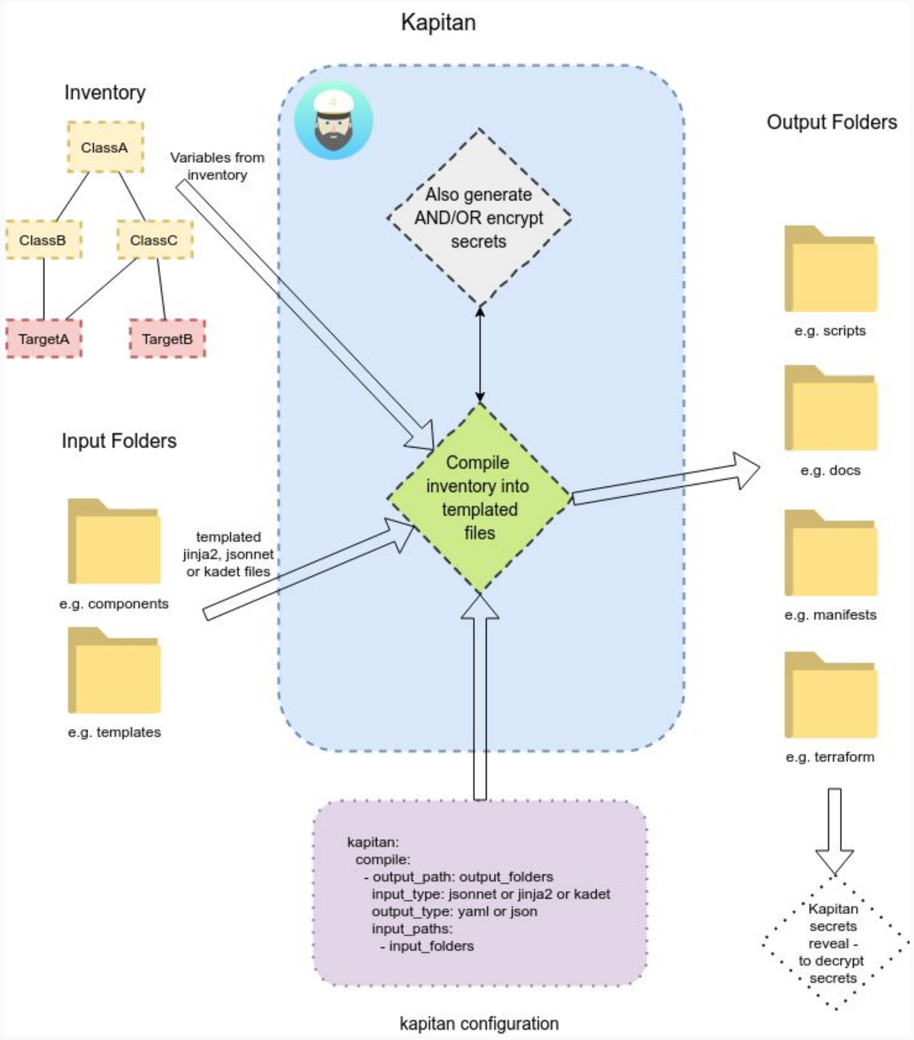
Present & Future

Ricardo Amaro
@ramaro

Present

Overview

High level Overview



```
kapitan:
  compile:
    - output_path: output_folders
      input_type: jsonnet or jinja2 or kadet
      output_type: yaml or json
      input_paths:
        - input_folders
```

Templates

```
├── templates
│   ├── components
│   │   ├── coffee_shop
│   │   │   ├── main.jsonnet          ┐
│   │   │   ├── secret.jsonnet        │
│   │   │   ├── service.jsonnet       ├ jsonnet
│   │   │   └── statefulset.jsonnet   ┘
│   │   ├── namespace
│   │   │   └── main.jsonnet
│   │   ├── nginx                     ┐
│   │   │   ├── __init__.py           ├ kadet
│   │   │   └── proxy.py              ┘
│   │   └── scripts                   ┐
│   │       ├── README.md             ├ jinja2
│   │       └── apply.sh              ┘
...
```

Compile

Templates + inventory + secrets = compiled/

Compile - all targets

```
$ kapitan compile
```

Compile - specific targets

```
$ kapitan compile -t dev-monitoring dev-apps ...
```

Inventory

Targets + classes

merges parameters in classes and targets

Inventory

```
├── inventory
│   ├── classes
│   │   ├── cluster                          ┐
│   │   │   ├── us.yml                       │
│   │   │   └── europe.yml                   ├ cluster classes
│   │   ├── common.yml                       ┘
│   │   └── component                        ┐
│   │       ├── coffee_shop.yml              │
│   │       ├── scripts.yml                  ├ component classes
│   │       ├── namespace.yml                │
│   │       └── nginx.yml                    ┘
│   └── targets                              ┐
│       ├── dev-apps.yml                     │
│       ├── dev-monitoring.yml               ├ targets import classes
│       └── qa-apps.yml                      ┘
...
```

Inventory - all targets

```
$ kapitan inventory
```

Inventory - specific targets

```
$ kapitan inventory -t dev-apps
```

Inventory - target from template - jsonnet

```
// components/coffee_shop/main.jsonnet

local kap = import "lib/kapitan.libjsonnet";
local inv = kap.inventory();

{
	name: inv.parameters.coffee_shop.name,
	port: inv.parameters.coffee_shop.port_number,
	...
```

Inventory - target from template - jinja2

```
{# components/coffee_shop/README.md #}

{% set inv = inventory.parameters %}

# About
This Coffee Shop instance is running on cluster {% inv.cluster.name %}
under namespace {% inv.target %}

# Connecting
You can reach this app on url {% inv.coffee_shop.url %}
...
```

Inventory - all targets from jinja2

```
{% for target in inventory_global %}
hello {{ target }}
{% endfor %}
```

Inventory - all targets from jsonnet

```
local inv_global = kap.inventory_global();

{
        [target_name + ".yml"]: { foo: "bar" },

}
for target_name in std.objectFields(inv_global)
```

Secrets

Securely store sensitive data

access secrets from command line, inventory & templates

Secrets

```
{
        user: "john_doe”,
        password: "?{gkms:dev/coffee_shop-pass}",
...
```

Secrets - command line

```
$ kapitan secrets --write gkms:dev/coffee_shop-pass -f pass.txt
```

Secrets - dynamically from secret functions

```
{
    user: "barista”,
    password: "?{gkms:dev/coffee_shop-pass|randomstr:12}",
...
```

Secrets - from component (compiled)

```
---
user: barista
password: ?{gkms:dev/coffee_shop-pass:deadbeef}
...
```

Secrets - from component (revealed)

```
$ kapitan secrets --reveal -f compiled/dev-apps/coffee_shop_creds.yml

---
user: barista
password: Gm9OGGHYEuT4JDe5K7WE!
...
```

Secrets - backends

?{**gkms**:dev/coffee_shop-pass|randomstr:12}

?{**awskms**:dev/coffee_shop-pass|randomstr}

?{**gpg**:dev/coffee_shop-pass|randomstr}

?{**ref**:dev/coffee_shop-pass|**randomstr:8**} # not encrypted

Functions - jsonnet

file_read()

yaml_load() / yaml_dump()

jinja2_render_file()

sha256_string()

gzip_b64()

inventory() / inventory_global()

Functions - jinja2 filters

sha256

yaml

b64encode / b64decode

fileglob

bool

to_datetime

strftime

regex_replace

regex_escape

...

Future

A planned and aspirational overview

KAPs - Kapitan proposals

https://github.com/deepmind/kapitan/tree/master/docs/kap_proposals

KAP-0 - Kadet: python input type

```python
class MyApp(BaseObj):
  def body(self):
    self.root.name = "myapp"
    self.root.inner.foo =
"bar"
    self.root.list = [1, 2, 3]
```

$\Rightarrow$

```yaml
---
name: myapp
inner:
  foo: bar
list:
  - 1
  - 2
  - 3
```

Compiles python into json/yaml/plain
Benefits from python ecosystem (modules, pip, etc)
Available today (experimental)

KAP-1 - External dependencies

```
parameters:
 kapitan:
  dependencies:
   - type: https
     output path: components/prometheus chart
     source: https://github.com/helm/charts/tree/master/stable/prometheus
```

Fetches from *https* and *git* sources into output_path
Supports subdirs, git commits, branches

Force with $ kapitan compile --fetch

KAP-2 - Helm charts input type

```
parameters:
 kapitan:
  compile:
   - input type: helm
     input path: components/prometheus_chart
     output_path: manifests
```

Compiles helm charts!

Interoperability with the inventory

Works with external dependencies (KAP-1)

KAP-3 - Schema validation

```
parameters:
  kapitan:
    validate:
       - output type: kubernetes.service
         version: 1.6.6
          output_path: manifests/my_app/service.yml
```

Validates manifests during compile
Kubernetes specific for now
Uses jsonschema

KAP-4 - Standalone executable

Create a portable/static binary or directory

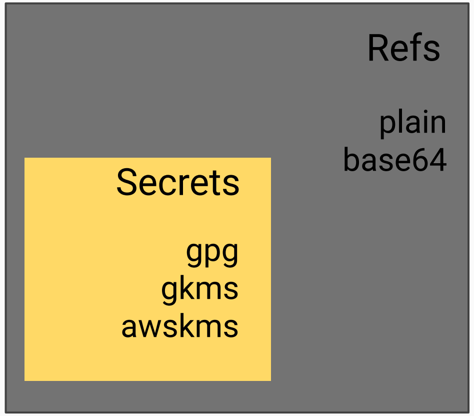
Cross-platform

Easier distribution and installation

KAP-5 - Kapitan ~~Secrets~~ Refs (references)

```
$ kapitan refs --write gpg:my/secret1 ...
$ kapitan refs --write base64:my/file ...
$ kapitan refs --write plain:my/info …

$ echo $USER | kapitan refs --write plain:my/user -f -
$ echo 'envoyproxy/envoy:v1.10' | kapitan refs --write plain:images/envoy -f -
```



Refs are more generic

Secrets are a sub type of Refs (encrypted)

Backend type representative of how data is stored: ~~?{ref:dev/coffee_shop-pass|randomstr:8}~~

Plain backend useful for updating values from the command line:

KAP-6 - Hashicorp Vault ref backend

```
$ kapitan refs --write vault:dev/coffee_shop-pass ...
```

?{**vault**:dev/coffee_shop-pass|randomstr:12}

New vault backend will write and store secrets into a Vault instance

KAP-?? - Target labels and selectors

```
parameters:
 target: dev-monitoring-1
 kapitan:
  labels:
   env: dev
   name: monitoring
```

```
kap.inventory_select({env: “dev”, name: “monitoring”});
$ kapitan compile -l env=dev,name=monitoring
```

Compile/Inventory all targets matching a selector

Removes the need to know all target names

KAP-?? - Extensions

Bring in your own jsonnet, jinja and kadet functions

Keep them in your templates

Questions?