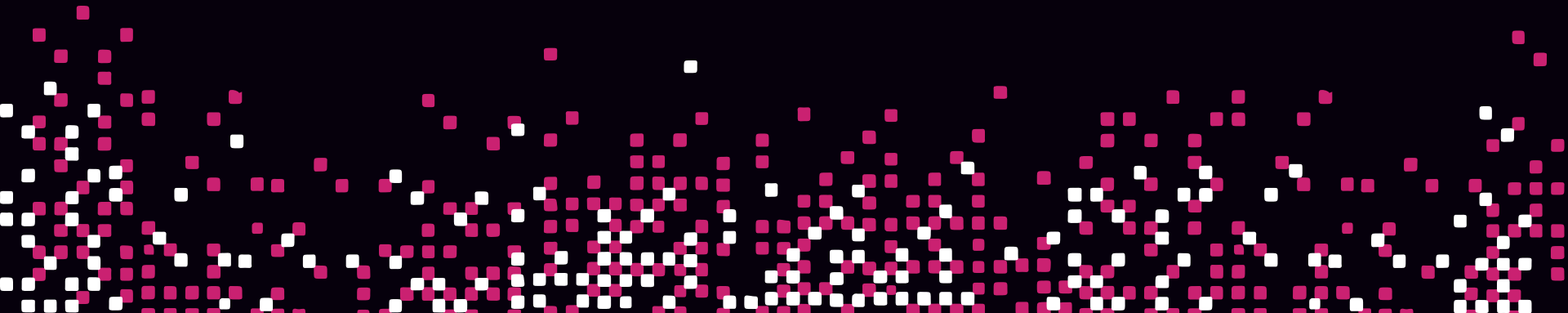




Access the Consul Key/Value



Access the Consul Key/Value

Objective 4a: Understand the capabilities and limitations of the KV store

Objective 4b: Interact with the KV store using both the Consul CLI and UI

Objective 4c: Monitor KV changes using watch

Objective 4d: Monitor KV changes using envconsul and consul-template

1

2

3

4

5

Difficulty Level



What is the Key/Value Store?

- Centralized Key/Value store that allows users to store objects
 - Distributed architecture – data is replicated across **all** server nodes
 - Installed with Consul and **always** enabled
 - Can store **any** type of data – no restriction on type of object
- Consul K/V Use Cases
 - Commonly used to store configuration parameters and related metadata
 - Accessed by the application at runtime
 - Accessible by both **server** and **client** agents



What the Key/Value Store Is Not?

- Not a full featured database like DynamoDB
- Not an encrypted K/V store – data is not encrypted
 - Store only non-sensitive data (use Vault instead)
- Does not have a directory structure
 - Does support using / to organize data, though
 - / is treated like any other character
 - Example: kv/app01/key, web/api/access, etc.
- Consul K/V data is stored in a single datacenter – no built-in replication between datacenters



Additional Information about Consul K/V

- Object size limitation: 512KB per object
- Backup and Recovery
 - Use `consul snapshot save` command
 - Use Consul snapshot agent ([Enterprise](#))

Lots more on backup
and recovery in
objective 5



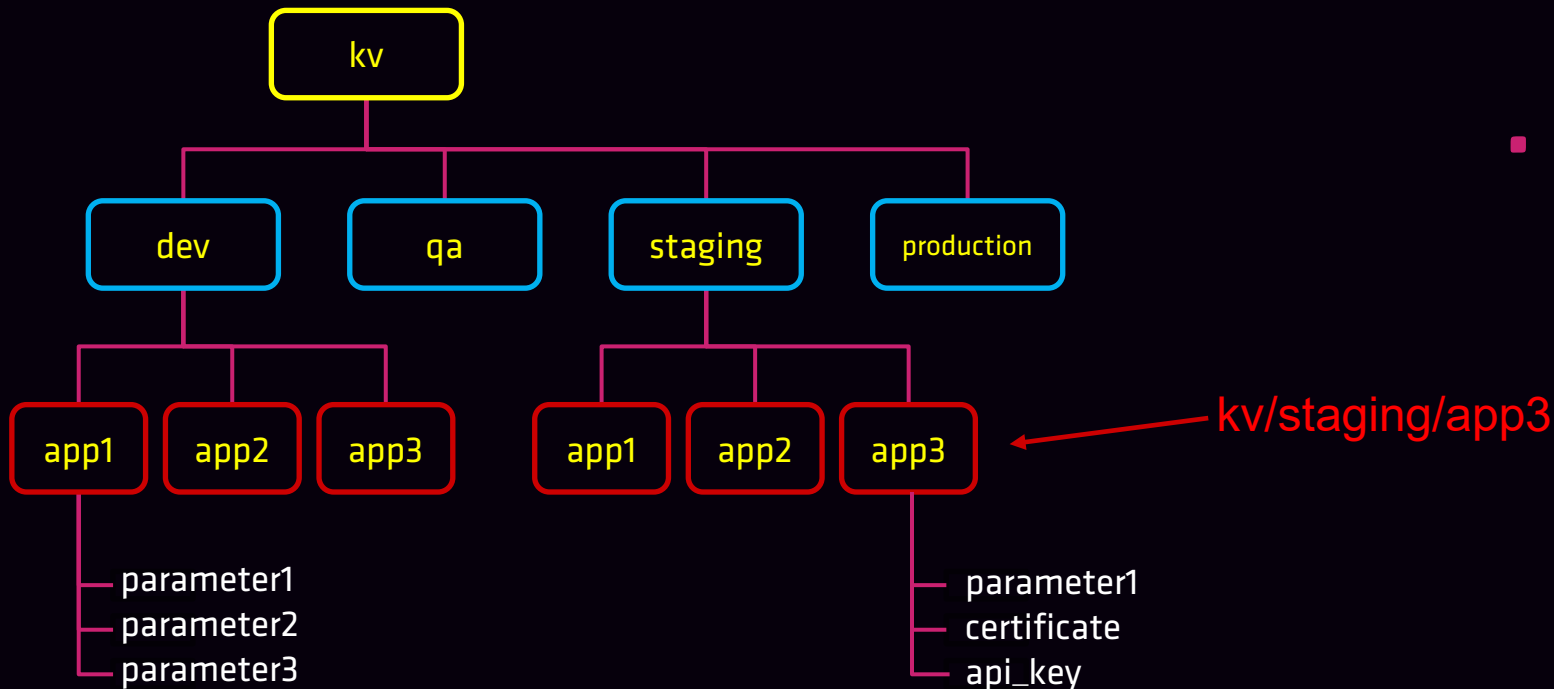
Designing the K/V Structure

- Design your K/V structure before you start using it
 - Sit down with your peers (**remote works as well**) and outline what the K/V structure will look like
 - Align the design to your application and infrastructure teams and operational use cases
- Every K/V structure will be different, and it should be designed for the **current** and **future** use cases



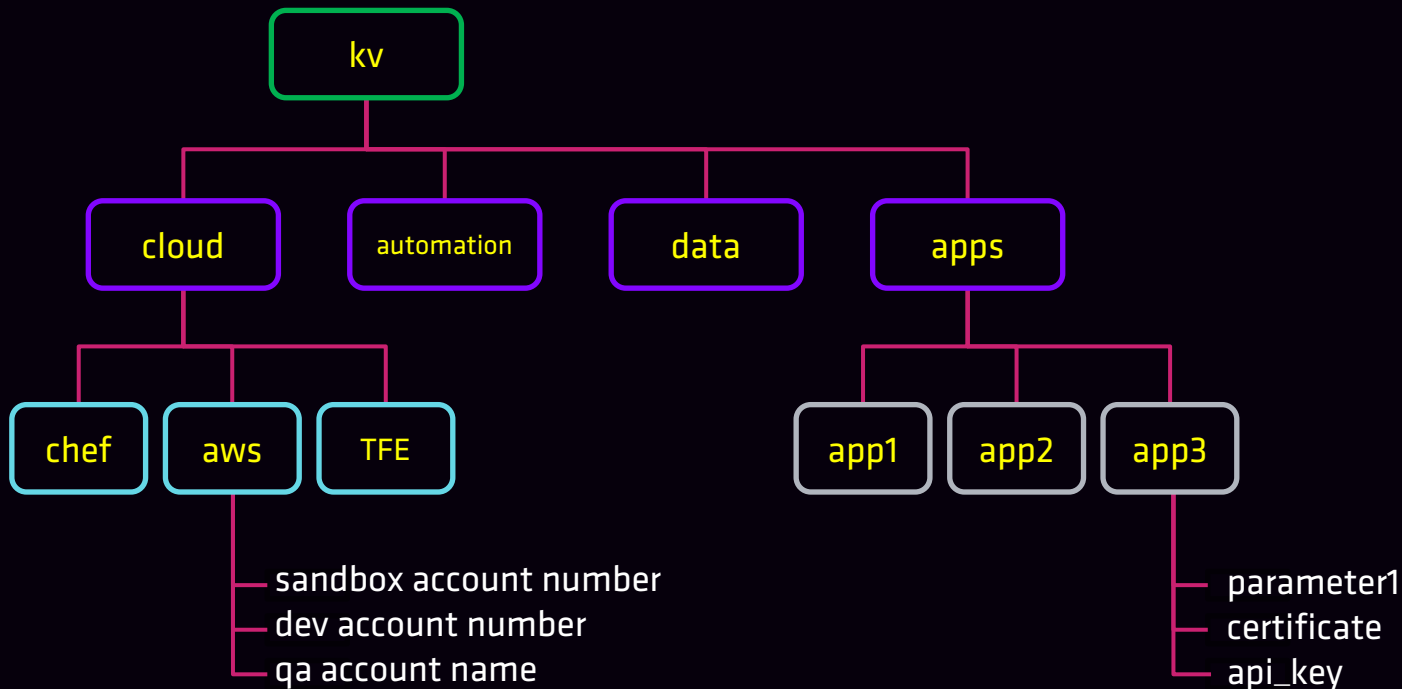
Designing the K/V Structure

Design based on application environment



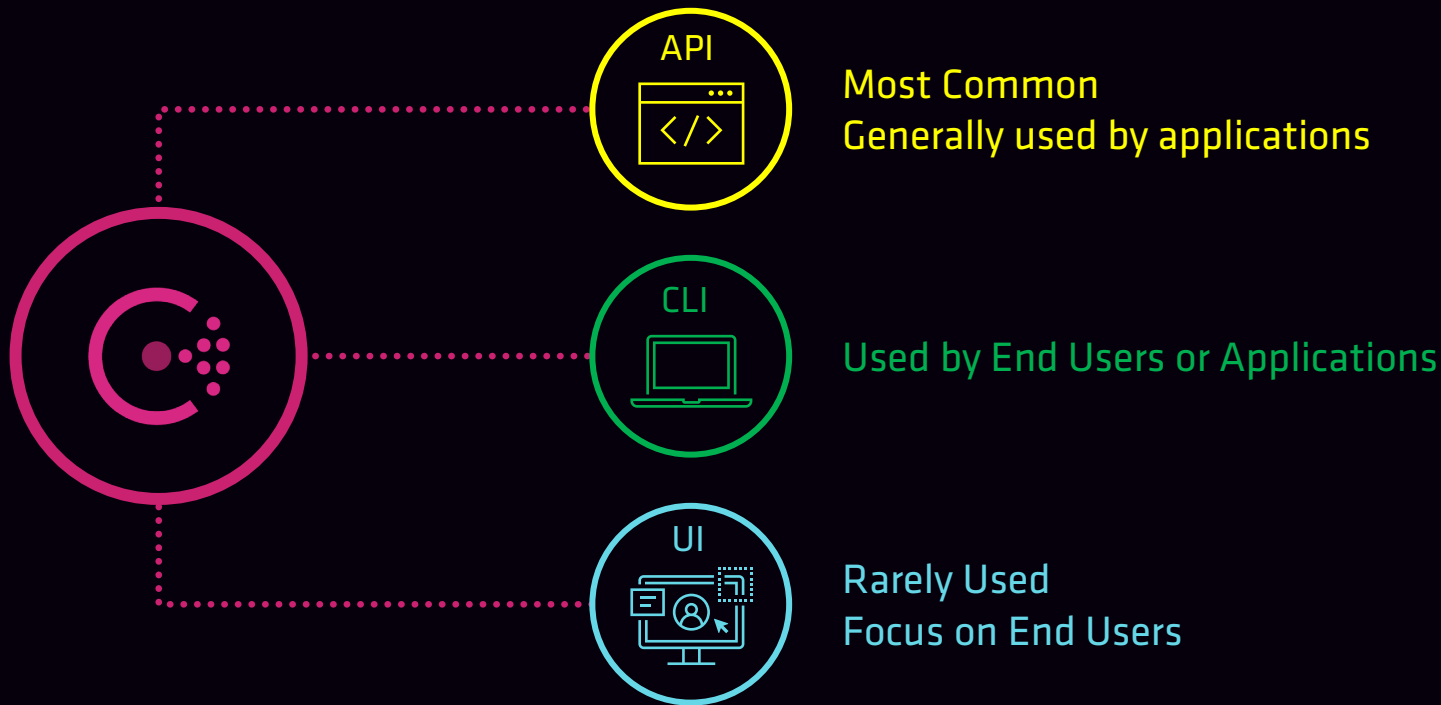
Designing the K/V Structure

Design based on teams



Accessing the Key/Value Store

Consul Interfaces



Accessing the Key/Value Store

HTTP API


- Uses the `/kv` endpoint
- Path will be created if it does not exist

Adding data to Consul K/V

- Method: `PUT`
- response value is either `true/false` on whether the create/update was successful

Terminal

```
$ curl --request PUT -data 'enabled' https://consul.example.com:8500/v1/kv/data/app4  
true
```



Accessing the Key/Value Store

HTTP API

Retrieving data from Consul K/V

- Method: GET
- response is base64 encoded (do not confuse this with encryption)

Terminal

```
$ curl https://consul.example.com:8500/v1/kv/data/app4 | jq
[
  {
    "LockIndex": 0,
    "key": "data/app4",
    "Flags": 0,
    "value": "J2VuYWJsZWQn",
    "CreateIndex": 69,
    "ModifyIndex": 87
  }
]
```

Terminal

```
$ echo "J2VuYWJsZWQn" | base64 --decode
'enabled'
```



Accessing the Key/Value Store

Command-Line Interface (CLI)

- Uses the **consul kv** command

- **put**
- **get**
- **delete**
- **export**
- **import**

Terminal

```
$ consul kv put app1/config/apikey 4fe20s12a02$23  
Success! Data written to: app1/config/apikey
```

Terminal

```
$ consul kv get app1/config/apikey  
4fe20s12a02$23
```

Terminal

```
$ consul kv delete app1/config/apikey  
Success! Data deleted at key: app1/config/apikey
```



Accessing the Key/Value Store

User Interface (UI)

The screenshot shows a web interface for managing a Key/Value store. The top navigation bar includes tabs for 'dc1', 'Services', 'Nodes', 'Key/Value', 'ACL', and 'Intentions'. The 'Key/Value' tab is selected. Below the navigation, the breadcrumb path is '< Key / Values / app1 / config'. The main content area displays the key name 'apiKey' and its value '1 4fe20s12a02\$23'. A yellow arrow points from the text 'Key Value' to the value field. A red arrow points from the text 'View data as...' to a dropdown menu showing 'YAML'. At the bottom, there are 'Save', 'Cancel', and 'Delete' buttons. Annotations include a blue line pointing to the 'Key/Value' tab and a purple line pointing to the breadcrumb path.

Key/Value Tab

Key Name

Key Value

View data as...

Save Cancel Delete



Accessing the Key/Value Store

Limiting Access to the K/V

- To limit access to the K/V, use **Consul ACLs**
- Will protect access through all **three** interfaces

More information on ACLs can be found in Objective 8



Monitor KV Changes Using Watch

Watch provides a way to monitor for specific changes in Consul

- Built-in to Consul – no additional binary/configuration needed
- Once the "view" of data is updated, a specific handler is invoked...
- ...or just log it to STDOUT

Handlers

- Can invoke a **command** (using a shell) when change is detected
- Can also hit an **HTTP endpoint**



Monitor KV Changes Using Watch

Watch Types

Key	Watch a specific KV pair
keyprefix	Watch a prefix in the KV store
services	Watch the list of available services
nodes	Watch the list of nodes
service	Watch the instances of a service
checks	Watch the value of health checks
event	Watch for a custom user event



Monitor KV Changes Using Watch

Configuring a Watch

- Watches are implemented using blocking queries in Consul **API**
- Watches can be added to an **agent configuration**, causing it to be run once the agent is started...
- ...or it can be started outside of the agent using the **consul watch** command
- **Even if only one entry changed, Consul will return all the matching entries for the query**



Monitor KV Changes Using Watch

Watch Configurations

Configuration - Key Watch with Executable Handler

```
{  
  "type": "key",  
  "key": "prod/database/mysql",  
  "handler_type": "script",  
  "args": ["/usr/bin/update_database_creds.sh"]  
}
```

Annotations:

- Type of watch (points to "type": "key")
- Key to Watch (points to "key": "prod/database/mysql")
- Run this script (points to "args": ["/usr/bin/update_database_creds.sh"])

Command Line - Key Watch with Executable Handler

Terminal

```
$ consul watch -type=key -key=prod/database/mysql /usr/bin/update_database.sh
```



What is envconsul?

- Envconsul launches a subprocess to set environment variables from data retrieved from Consul (and Vault!)
- Separate binary that runs on the application server (Consul client)
- envconsul populates the ENV(s) and the application reads the ENV(s)
 - Apps no longer need to read config files with sensitive data in clear text
 - Retrieve data from the KV or about Consul services
- Allows **simplified application integration** without the application knowing it's using Consul to retrieve values



Using envconsul

Terminal

```
$ consul kv put db01/DB_ADDRESS 10.2.23.98  
$ consul kv put db01/DB_PORT 3306  
$ consul kv put db01/DB_MAX_CONNS 50
```

Populate Data

```
$ envconsul -prefix db01 env
```

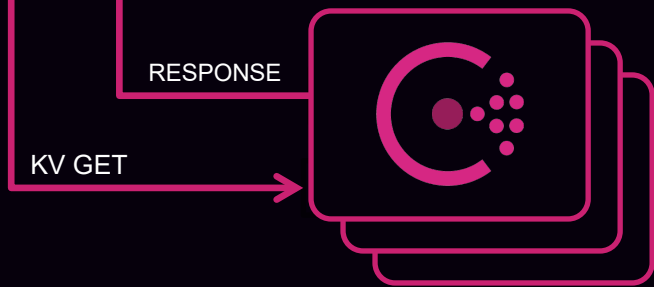
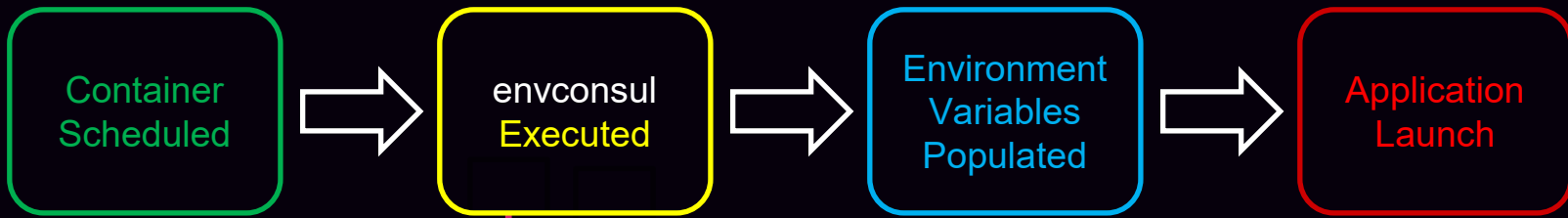
Set Environment Variables w/ envconsul

```
DB_ADDRESS=10.2.23.98  
DB_PORT= 3306  
DB_MAX_CONNS=50
```

} New Environment Variables



What is envconsul?



Consul Cluster



What is consul-template?

- consul-template populates values from Consul into the file system running the consul-template daemon
- Separate binary that runs on the application server (Consul client)
- Uses a preconfigured, templated file as an input
- **Outputs** a file with data populated from Consul



What is consul-template?

Terminal

config.yml.tmpl

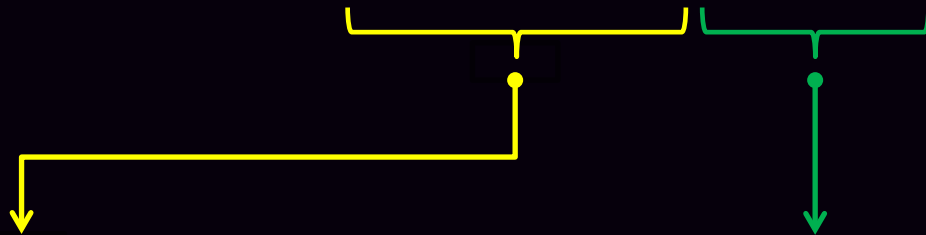
```
production:
  adapter: mysql
  encoding: unicode
  database: eCommerce
  host: mysql.service.consul
  {{ with secret "database/creds/mysql" }}
  username: "{{{ .Data.username }}"
  password: "{{{ .Data.password }}"
  {{ end }}
```

} Consul Template
Parameters



What is consul-template?

```
$ consul-template -template="config.yml.tpl:config.yml" -once
```

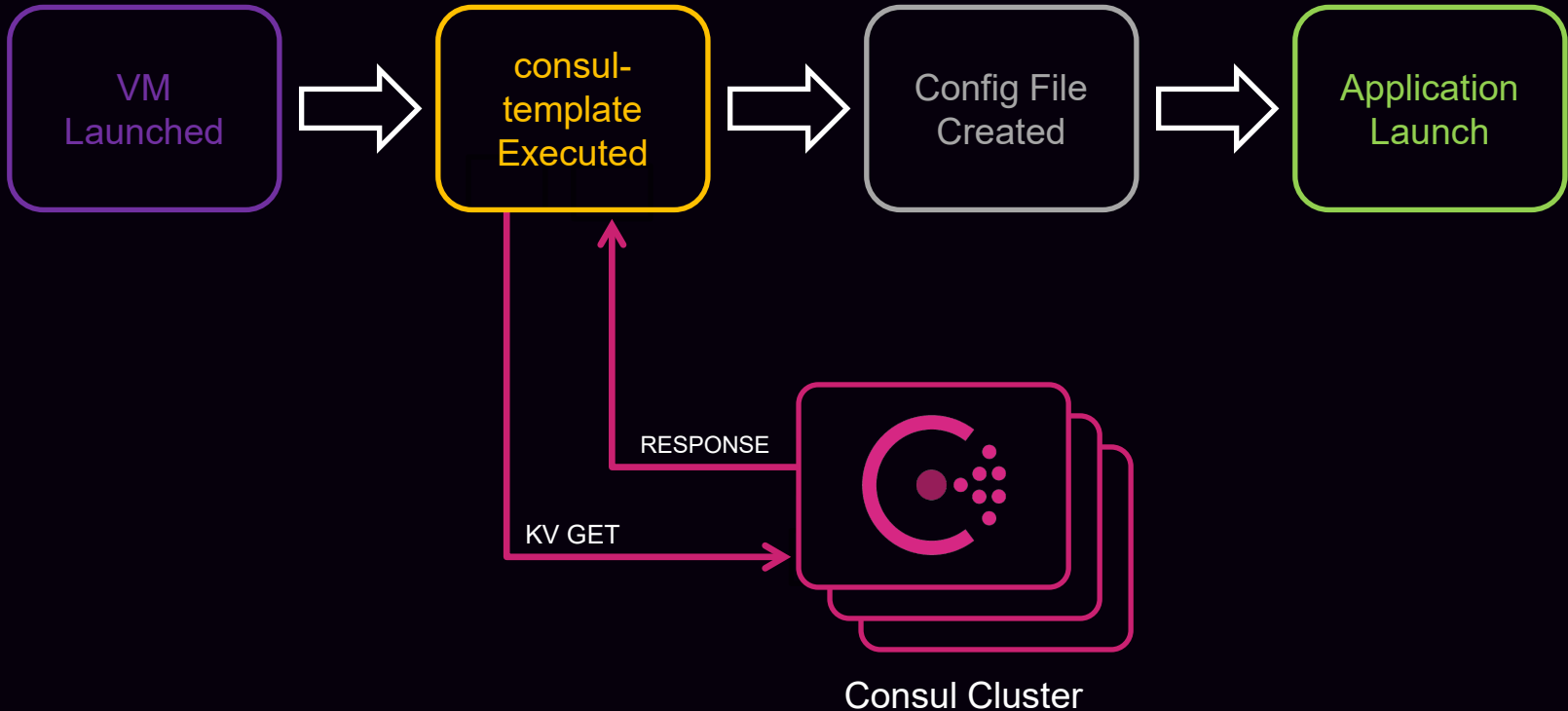


```
Terminal
config.yml.tpl
production:
adapter: mysql
encoding: unicode
database: eCommerce
host: mysql.service.consul
{{ with secret "database/creds/mysql" }}
username: "{{ .Data.username }}"
password: "{{ .Data.password }}"
{{ end }}
```

```
Terminal
config.yml
production:
adapter: mysql
encoding: unicode
database: eCommerce
host: mysql.service.consul
username: v-vault-mysql-u5433c02234caw4e
password: 1tq33s7z5uprpxxy68
```



What is consul-template?



Access the Consul Key/Value

Objective 4a: Understand the capabilities and limitations of the KV store

Objective 4b: Interact with the KV store using both the Consul CLI and UI

Objective 4c: Monitor KV changes using watch

Objective 4d: Monitor KV changes using envconsul and consul-template

1

2

3

4

5

Difficulty Level





END OF
SECTION

