KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon CloudNativeCon Europe 2022

GitOpsify Everything: When Crossplane Meets Argo CD

Ying Mo (moyingbj@cn.ibm.com)
Ken Murray (kenmurr3@ie.ibm.com)

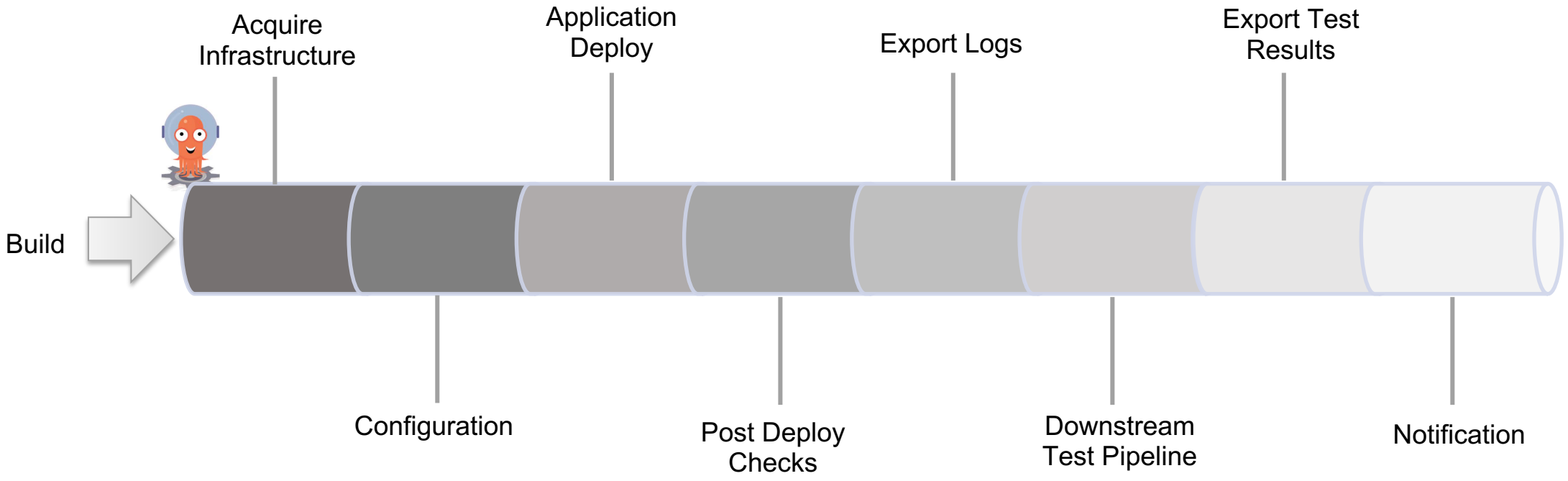
Agenda

- Adoption of Crossplane & Argo CD in CI/CD Pipelines
  - CI/CD Deployment Pipeline – Generic View
  - Infrastructure Provisioning / Lifecycle Management
  - Crossplane as an abstraction platform for Infrastructure Provisioning
  - Managing Infrastructure Lifecycle and Application Deployment using GitOps

- Extending Argo CD to Traditional IT System by Crossplane
  - Automation in Non-Cloud-Native World – Generic View
  - The Crossplane Solution: Ansible Provider
    - How It Works?
    - How to Use It?
  - Crossplane Provider for Ansible: Deep Dive
    - The Ansible Run Lifecycle
    - Best Practices for Ansible User
    - Comparing with Ansible Operator

CI/CD Deployment Pipeline – Generic View

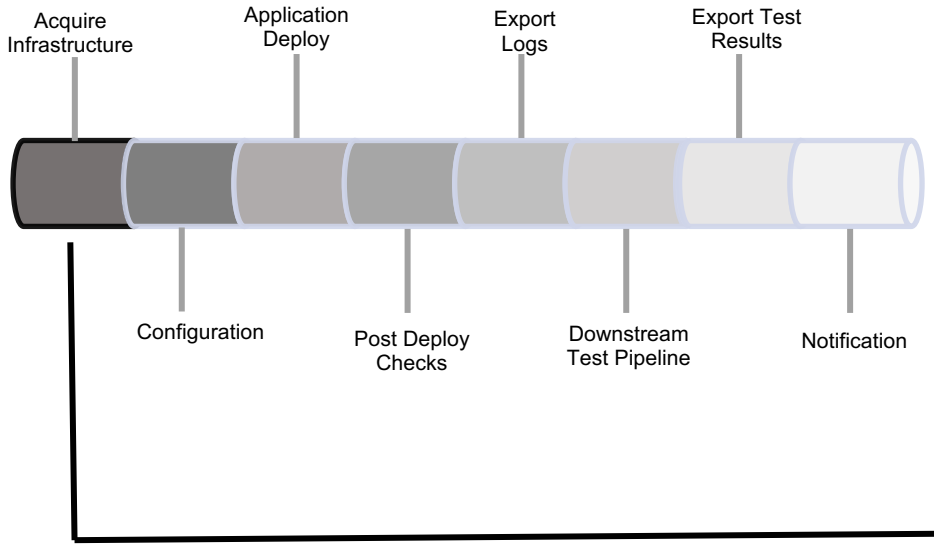
CI/CD Deployment Pipeline – Generic View

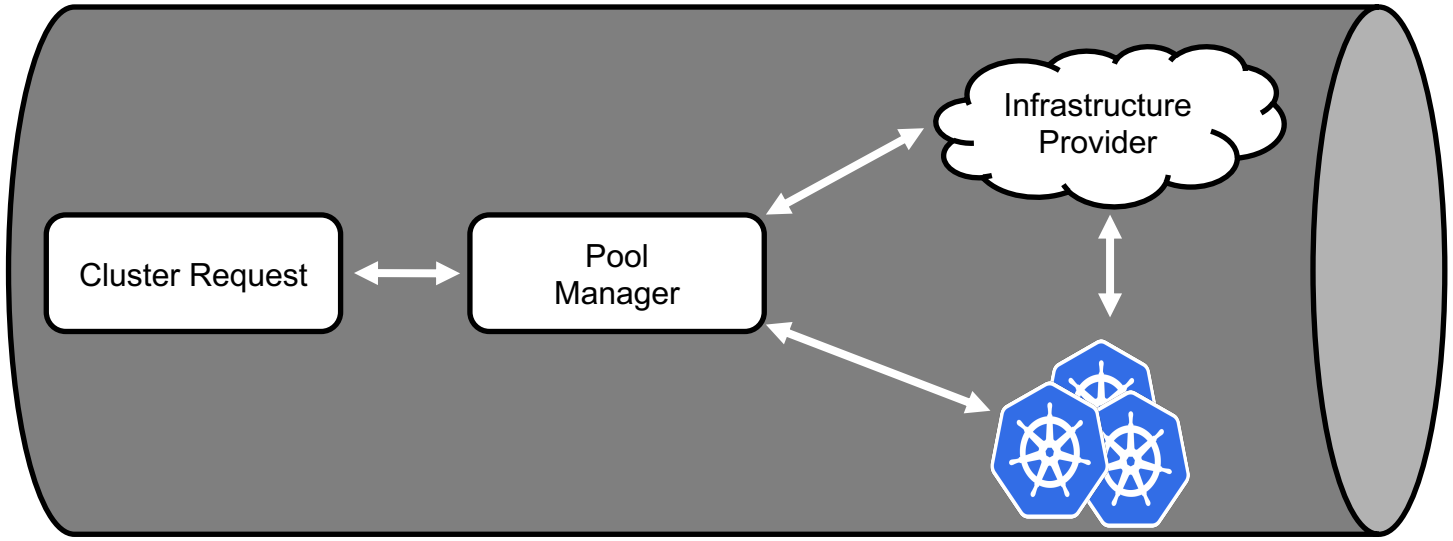


- Pipeline stages typically viewed as *"GitOps" ready* focus on Configuration & Application Deployment
- Infrastructure Management => Custom Infrastructure specific APIs

Infrastructure Provisioning / Lifecycle Management

Infrastructure Provisioning / Lifecycle Management



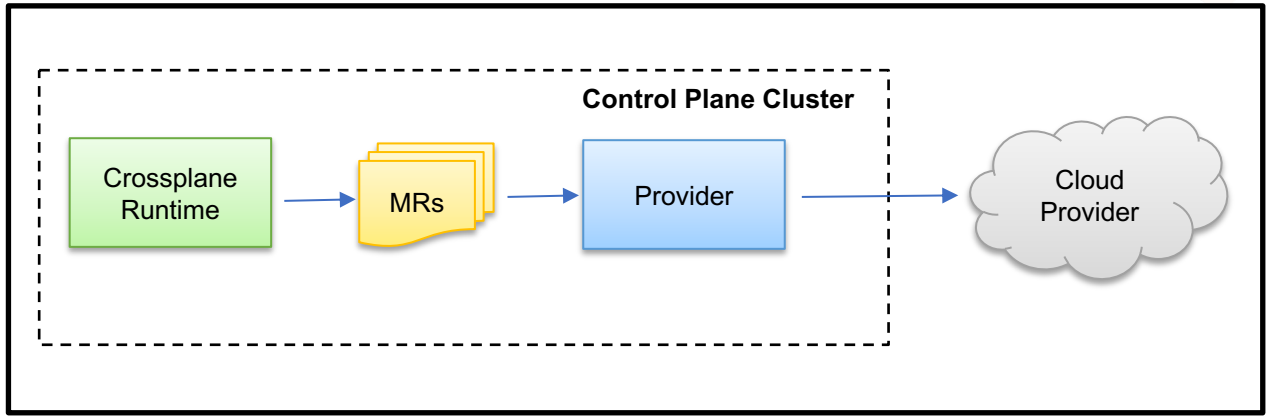


- Cluster Pool Characteristics
  - Set of provisioned OCP clusters
  - Managed cluster pool level
  - Supports variation on Cluster Specifications
  - Interface to Infrastructure Provider API for Cluster Provisioning

Crossplane as an abstraction platform for Infrastructure Provisioning

Crossplane as an abstraction platform for Infrastructure Provisioning

- Custom Crossplane Provider for OCP Cluster Provisioning




```
spec:
  forProvider:
    cluster_name: bvt-06143541
    ocp_version: 4.10.3
    platform: x
    product_group_id: "123"
    quota_type: product_group
    site: xyz
    worker:
      additional_disk: 250
      count: "9"
      cpu: "16"
      memory: "32"
  providerConfigRef:
    name: ocp-providerconfig
  writeConnectionSecretToRef:
    name: bvt-06143541-ocp-secret
    namespace: crossplane-system
```

- Simple cluster definition exposed via OCP resource
- Complexities of cluster provisioning masked from the user
- Cluster endpoints / credentials written to secrets on the Control Plane cluster
- Cluster status available in OCP resources

Crossplane as an abstraction platform for Infrastructure Provisioning

_cluster.tpl

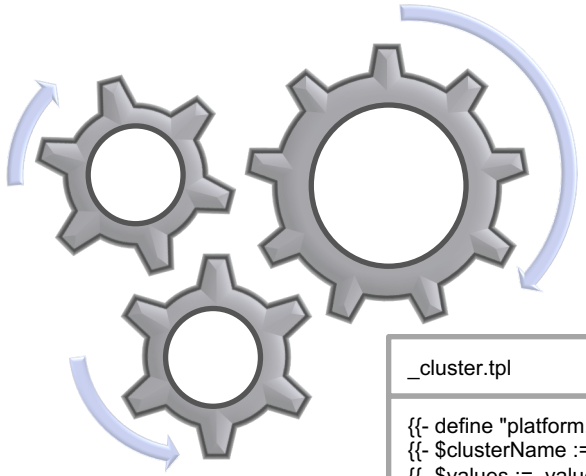
```
{{- define "platform.ocp" -}}
{{- $clusterName := .clusterName -}}
{{- $values := .values -}}
apiVersion: platform.crossplane.io/v1alpha1
kind: OCP
metadata:
  name: {{ $clusterName }}
spec:
  forProvider:
    cluster_name: {{ $clusterName }}
    site: {{ $values.spec.site }}
    product_group_id: "{{$values.spec.groupId}}"
    ocp_version: {{$values.spec.ocpVersion}}
    platform: "x"
    quota_type: product_group
    worker:
      count: "{{$values.spec.worker.count}}"
      cpu: "{{$values.spec.worker.cpu}}"
      memory: "{{$values.spec.worker.memory}}"
      additional_disk: {{$values.spec.worker.disk}}
  providerConfigRef:
    name: ocp-providerconfig
  writeConnectionSecretToRef:
    name: {{ $clusterName }}-ocp-secret
    namespace: crossplane-system
---
{{- end -}}
```

- Helm allows for the extraction of customizable configuration out of the managed resource and stored in ***values.yaml*** files
- Use of a Helm named template allows for passing of ***attribute values*** across cluster pool cycles

Crossplane as an abstraction platform for Infrastructure Provisioning

values.yaml

```
---
spec:
ocpVersion: "4.10.3"
site: "xyz"
groupId: "123"
worker:
count: "9"
cpu: "16"
memory: "32"
disk: "250"
```



cluster-pool.yaml

```
{{- include "platform.ocp"
(dict "clusterName" "bvt-06143543"
"values" .Values) -}}
```

_cluster.tpl

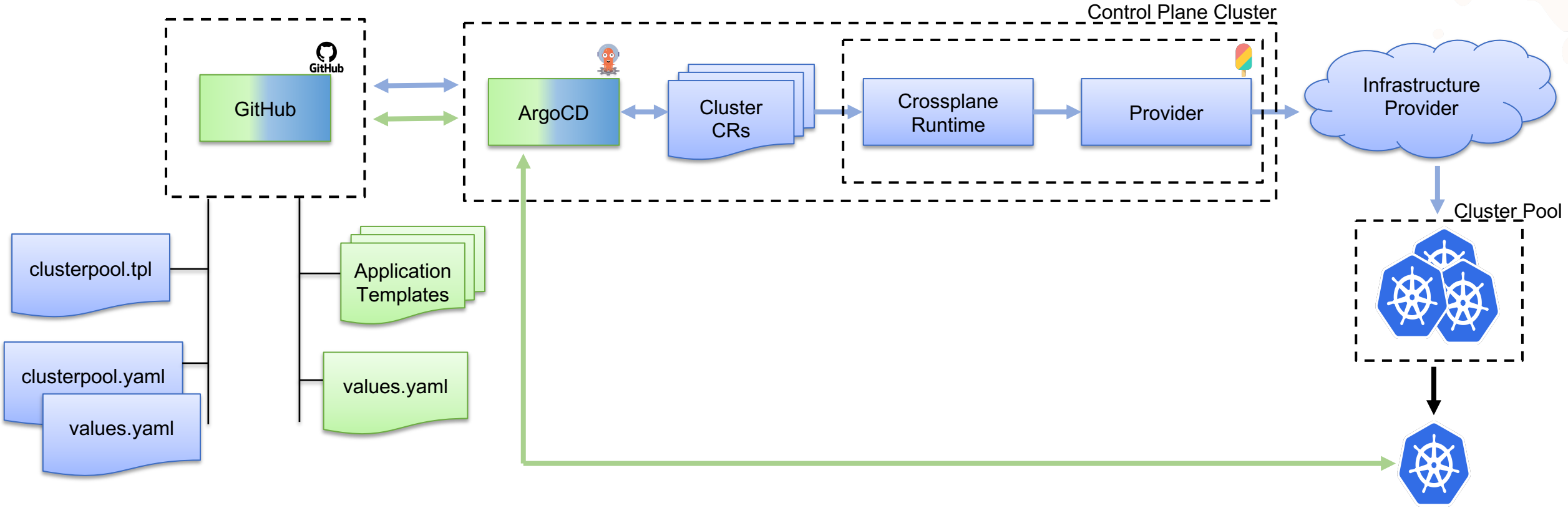
```
{{- define "platform.ocp" -}}
{{- $clusterName := .clusterName -}}
{{- $values := .values -}}
apiVersion: platform.crossplane.io/v1alpha1
kind: OCP
metadata:
  name: {{ $clusterName }}
spec:
  forProvider:
    cluster_name: {{ $clusterName }}
    site: {{ $values.spec.site }}
    product_group_id: "{{$values.spec.groupId}}"
    ocp_version: {{$values.spec.ocpVersion}}
    platform: "x"
    quota_type: product_group
    worker:
      count: "{{$values.spec.worker.count}}"
      cpu: "{{$values.spec.worker.cpu}}"
      memory: "{{$values.spec.worker.memory}}"
      additional_disk: {{$values.spec.worker.disk}}
  providerConfigRef:
    name: ocp-providerconfig
  writeConnectionSecretToRef:
    name: {{ $clusterName }}-ocp-secret
    namespace: crossplane-system
---
{{- end -}}
```

ocp-inst.yaml

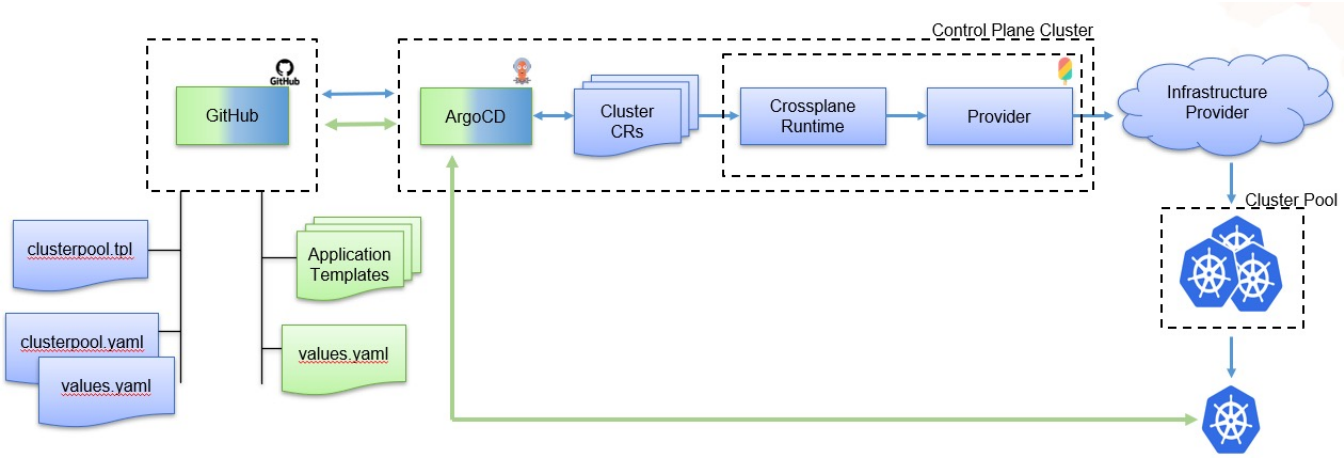
```
apiVersion: platform.crossplane.io/v1alpha1
kind: OCP
metadata:
  name: bvt-06143543
spec:
  forProvider:
    cluster_name: bvt-06143543
    site: svl
    product_group_id: 123
    ocp_version: 4.10.3
    platform: "x"
    quota_type: product_group
    worker:
      count: 9
      cpu: 16
      memory: 32
      additional_disk: 250
  providerConfigRef:
    name: ocp-providerconfig
  writeConnectionSecretToRef:
    name: bvt-06143543-ocp-secret
    namespace: crossplane-system
```

Managing Infrastructure Lifecycle and Application Deployment using GitOps

Managing Infrastructure Lifecycle and Application Deployment using GitOps



Managing Infrastructure Lifecycle and Application Deployment using GitOps

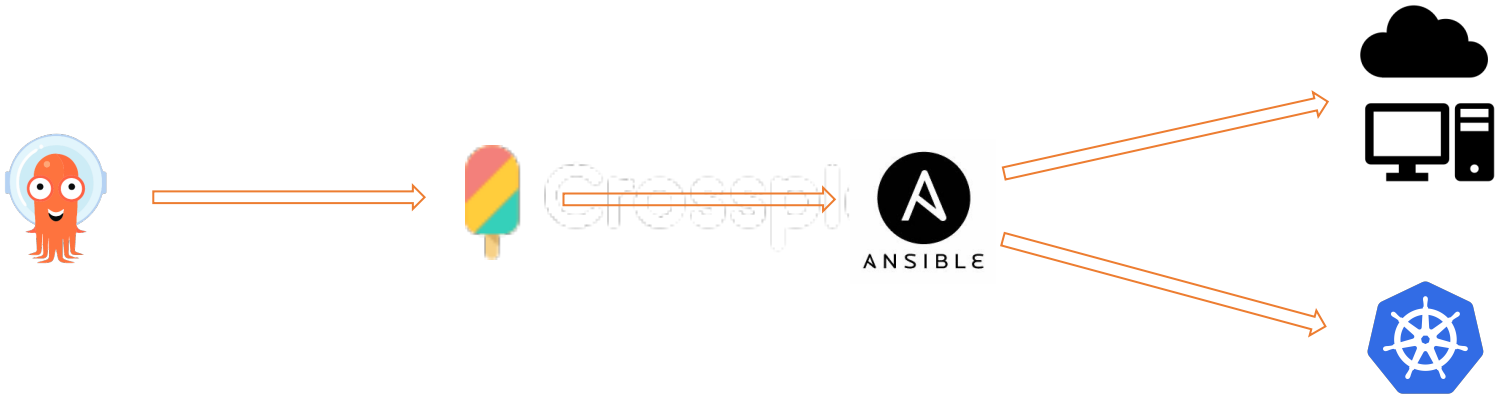


- Common GitOps approach for Infrastructure and Application Management
- Infrastructure deployment are versioned and audited
- Single source of truth of deployed infrastructure; avoid configuration drift
- Cluster state continuously reconciled from Git

Extending Argo CD to Traditional IT System by Crossplane

Automation in Non-Cloud-Native World – Generic View

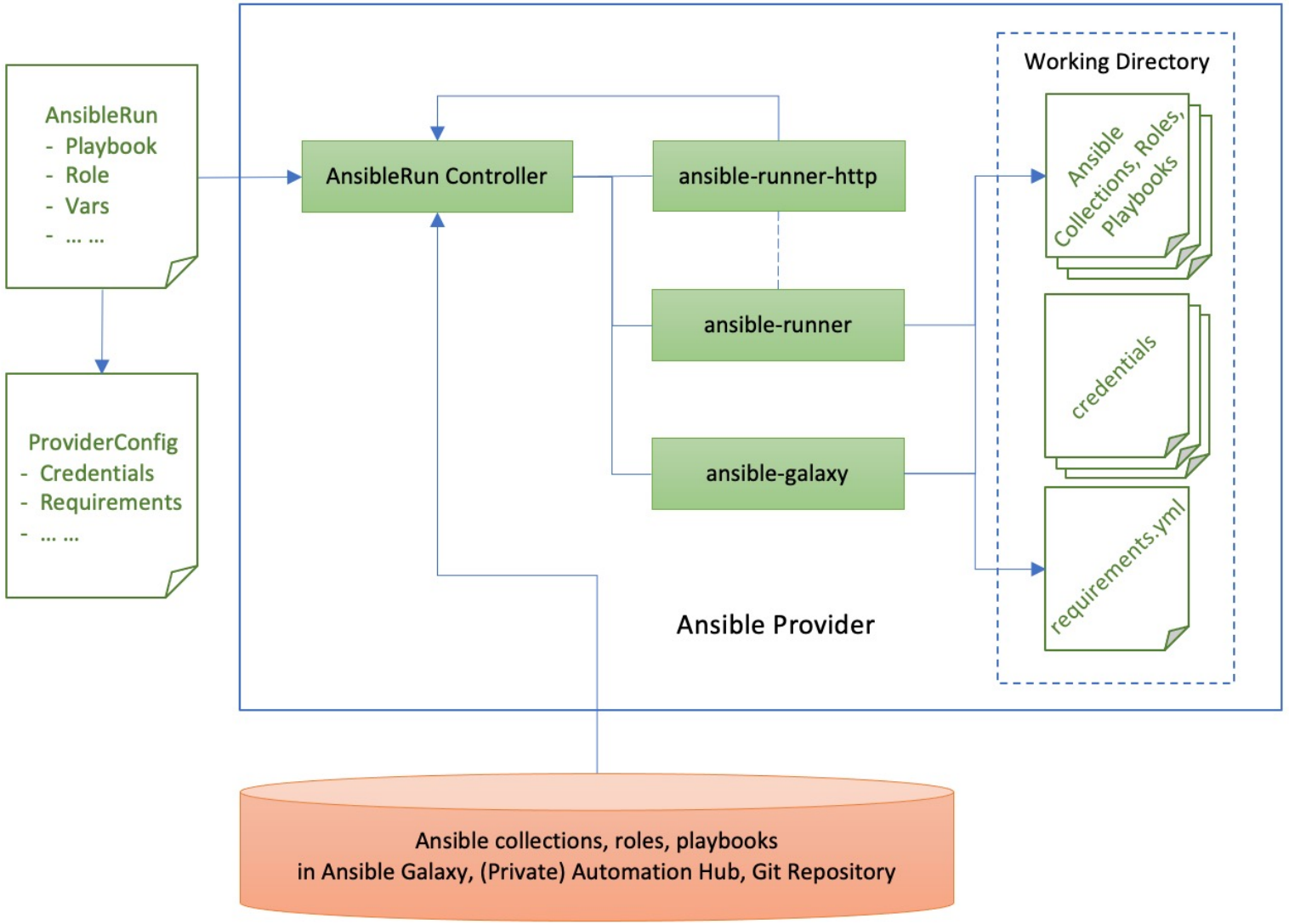
- Both Argo CD and Crossplane is designed to run in Kubernetes. This may not be true for real-world organizations who have traditional IT systems.
- Ansible, as a popular automation technology, is widely adopted by many of such organizations to automate management of their systems.
- The Crossplane provider for Ansible: a solution to drive management automation for both cloud native and non cloud native systems using the same way.



https://github.com/multicloudlab/crossplane-provider-ansible

The Crossplane Solution: Ansible Provider

How It Works?


AnsibleRun
- Playbook
- Role
- Vars
- ... ...
ProviderConfig
- Credentials
- Requirements
- ... ...
AnsibleRun Controller
ansible-runner-http
ansible-runner
ansible-galaxy
Ansible Provider
Working Directory
Ansible Collections, Roles, Playbooks
credentials
requirements.yml
Ansible collections, roles, playbooks
in Ansible Galaxy, (Private) Automation Hub, Git Repository

How to Use It?

Call remote role

```yaml
apiVersion: ansible.crossplane.io/v1alpha1
kind: AnsibleRun
metadata:
  name: remote-example
spec:
  forProvider:
    role: sample_namespace.sample_role
  providerConfigRef:
    name: provider-config-example
```

Call remote playbook

```yaml
apiVersion: ansible.crossplane.io/v1alpha1
kind: AnsibleRun
metadata:
  name: remote-example
spec:
  forProvider:
    playbook: sample_namespace.sample_collection.sample_playbook
  providerConfigRef:
    name: provider-config-example
```

```yaml
apiVersion: ansible.crossplane.io/v1alpha1
kind: ProviderConfig
metadata:
  name: provider-config-example
spec:
  credentials:
    - filename: .git-credentials
      source: Secret
      secretRef:
        namespace: crossplane-system
        name: git-credentials
        key: .git-credentials
  requirements: |
    ---
    collections:
      # Install a collection from Ansible Galaxy.
      - name: sample_namespace.sample_collection
        version: 0.1.0
        source: https://galaxy.ansible.com
      # Install a collection from GitHub repository.
      - name: https://github.com/sample_namespace/sample_collection.git
        version: 0.1.0
        type: git
```

Declare requirements

Pass variables

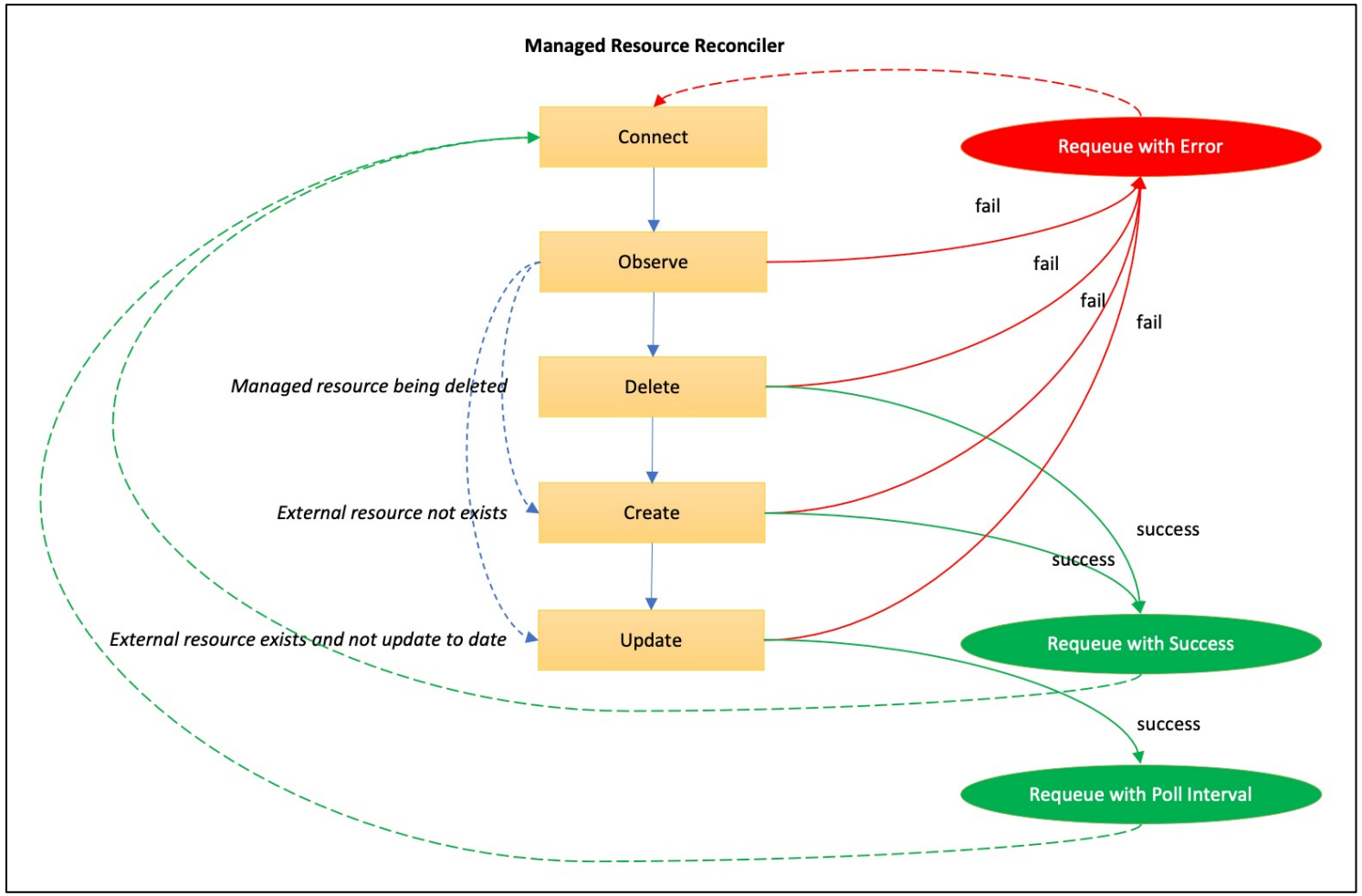
```yaml
apiVersion: ansible.crossplane.io/v1alpha1
kind: AnsibleRun
metadata:
  name: remote-example
spec:
  forProvider:
    role: sample_namespace.sample_role
    vars:
      foo: value1
      bar:
      - value2
      - value3
      baz:
        field 1: value1
        field 2: value2
  providerConfigRef:
    name: provider-config-example
```

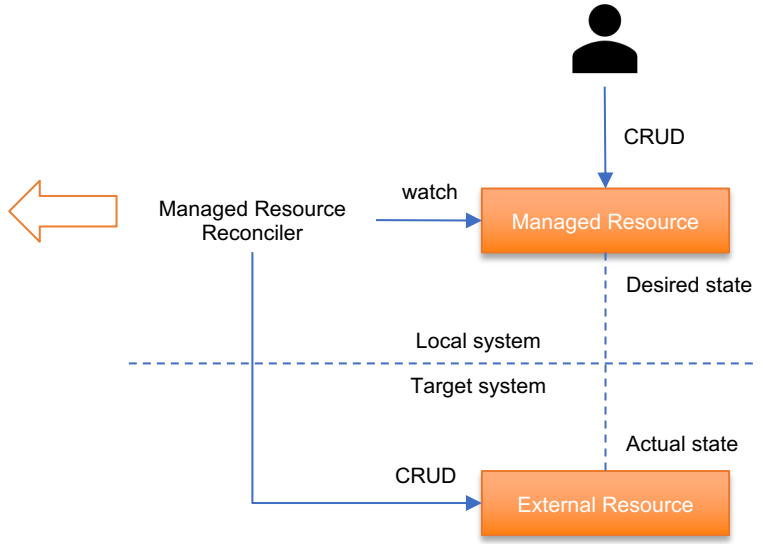
```yaml
apiVersion: ansible.crossplane.io/v1alpha1
kind: AnsibleRun
metadata:
  name: inline-example
spec:
  forProvider:
    # For simple cases, you can use an inline source to specify the content of
    # playbook.yml as opaque and inline yaml.
    playbookInline: |
      ---
      - hosts: localhost
        tasks:
          - name: simple-playbook
            debug:
              msg: You are running 'simple-playbook'
  providerConfigRef:
    name: provider-config-example
```

Define inline playbook

Crossplane Provider for Ansible: Deep Dive

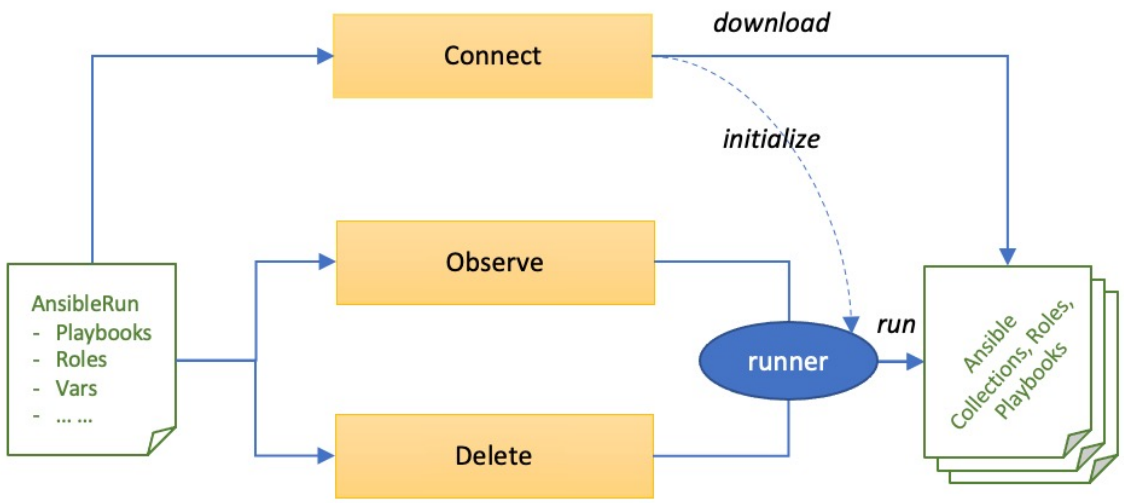
The Ansible Run Lifecycle: Crossplane Managed Resource Lifecycle



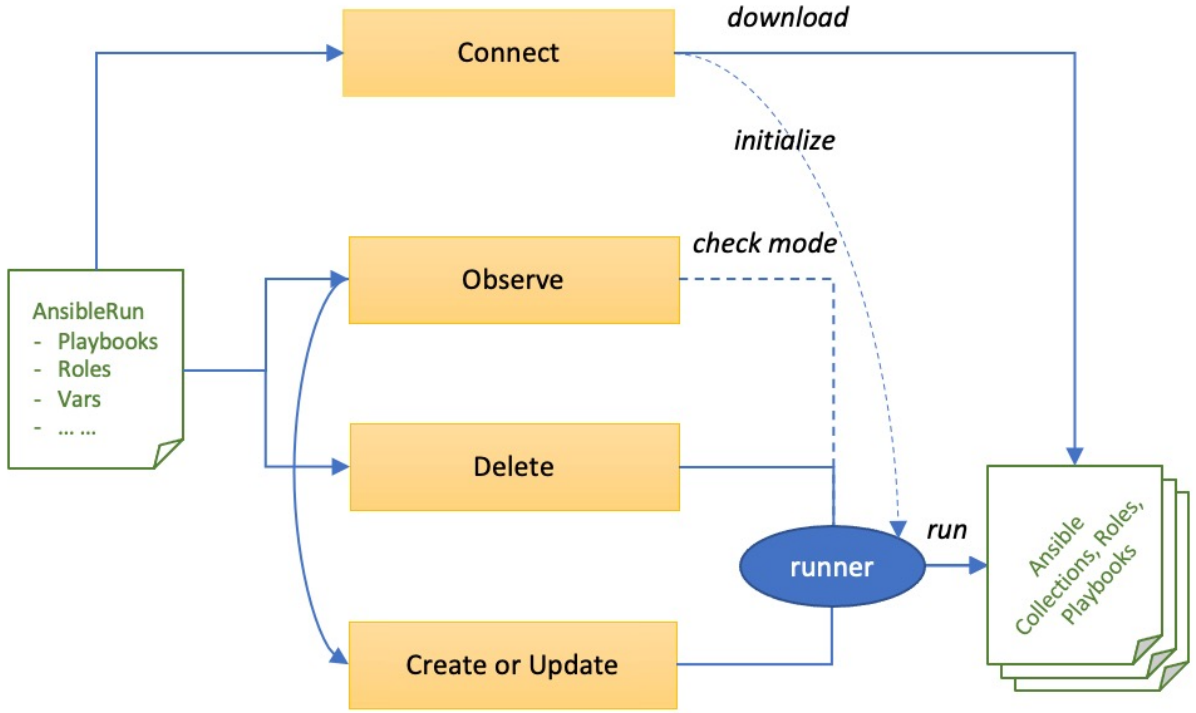


The Ansible Run Lifecycle: Mapping Ansible Run to Managed Resource Lifecycle

- Use Observe And Delete



- Run Check When Observe



Best Practices for Ansible User

- Writing Idempotent Roles or Playbooks

It is always a best practice to write Ansible roles or playbooks in an idempotent way. For Ansible provider, this is required because the same Ansible contents will be run many times in Managed Resource Lifecycle.

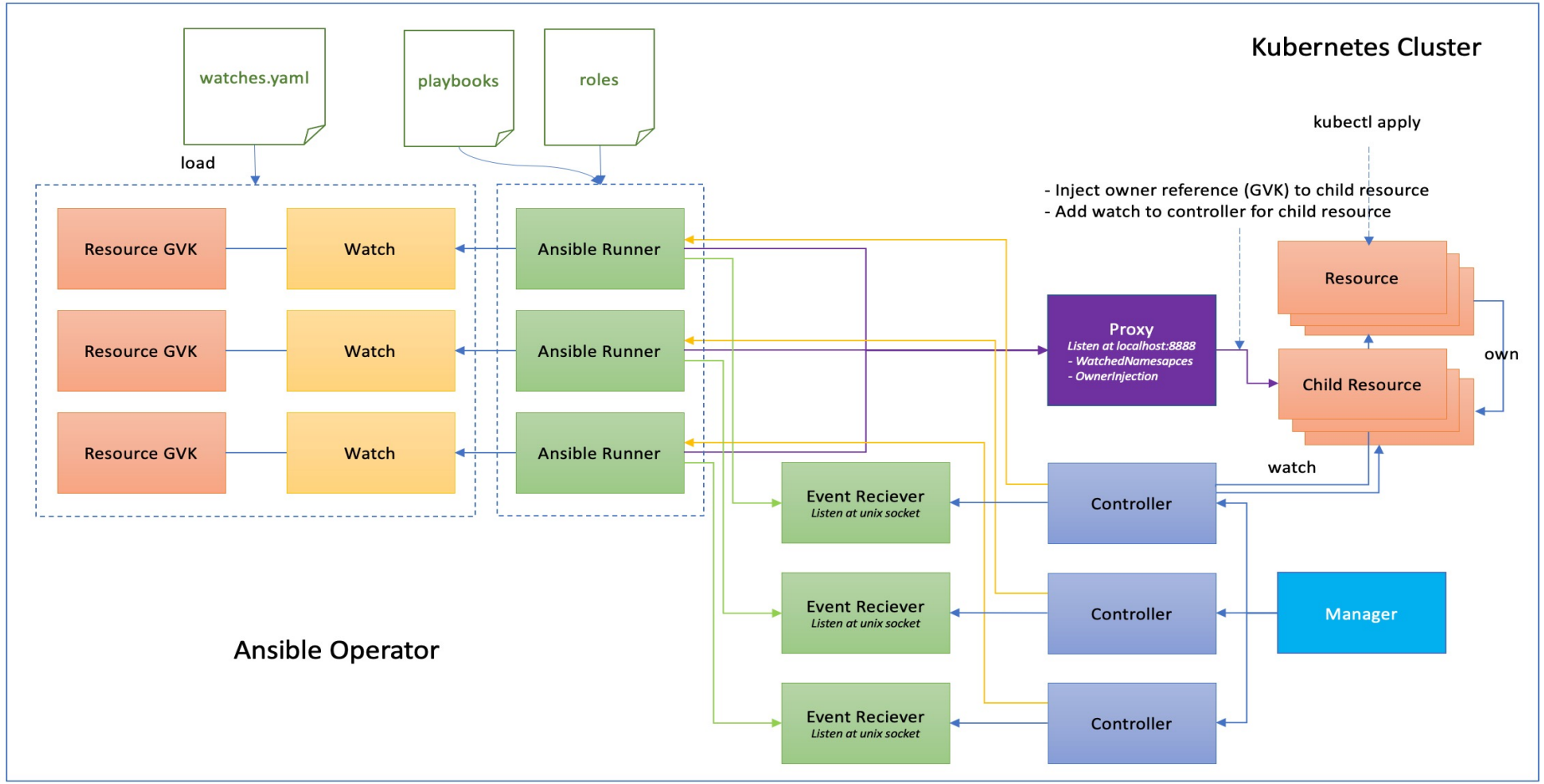
- Running Roles or Playbooks Per State

It is a common practice in many Ansible modules that support state field and behave differently according to the state value. For Ansible provider, this is required because the same Ansible contents will be used to handle both the case when the AnsibleRun resource is present and absent.

Comparing with Ansible Operator

https://sdk.operatorframework.io/docs/building-operators/ansible/

- Design for Kubernetes vs. design for both Kubernetes and non-Kubernetes.
- Bundle Ansible contents into container image at build time vs. prepare Ansible contents at runtime.
- Operate on in-cluster resources vs. operates on the remote resources.



Welcome Contribution to Crossplane Provider for Ansible!

Design Document: https://github.com/multicloudlab/crossplane-provider-ansible/blob/main/docs/design.md