GitOps, Jenkins X & the Future of CI/CD

Tracy Miranda, Director of Open Source Community | CloudBees

tmiranda@cloudbees.com | @tracymiranda
So, DevOps…

Almost want to yell - Stop saying DevOps - it doesn't mean what you think it means.

2:33 AM - 13 Apr 2017

5 Retweets 10 Likes
DevOps is the new legacy
Cloud Native Technologies

- On demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service
DevOps -> Cloud
GitOps -> Cloud Native
What would CI/CD look like if we had unlimited, scalable resources?
DORA State of DevOps Reports

2018

Accelerate: State of DevOps
Strategies for a New Economy

@tracymiranda
The Science of Lean Software and DevOps

“software delivery is an exercise in continuous improvement, and our research shows that year over year the best keep getting better, and those who fail to improve fall further and further behind.”

- Nicole Forsgren
Building and scaling high performance technology organisations

**Throughput**
- Frequent deployments
- Low lead time from commit to deploy

**Stability**
- Fast mean time to recovery (MTTR)
- Low change failure rate
GitOps

Operation by pull request

Git as the single place where we operate

All changes are observable
The GitOps Effect: Delivering Quality at Speed

2 → 150+
deployments per week

Via @fractallambda ref: https://twitter.com/tracymiranda/status/1045693846464729094
Git as the single source of truth of a system's desired state

GitOps Diffs compare desired state with observed state (eg Kubediff, Terradiff, Canary...)

All intended operations are committed by pull request, for all environments

All diffs between Git and observed state lead to (auto) convergence using tools like K8s

All changes are observable, verifiable and audited indisputably, with rollback & D/R
Best Practices for CI/CD for cloud native apps

1. Figure out the best practices of how to CD cloud native apps
   • Not just build and test, but review, promote, changelog, collaborate, etc.

2. Integrate best of breed software in this ecosystem to achieve it

3. Kubernetes as foundation & means to an end

4. Multicloud: Democratize it by building a high level, pleasant CLI

5. GitOps: deployments should be recorded and tracked in Git
Best Practices for CI/CD for cloud native apps

1. Figure out the best practices of how to CD cloud native apps
   • Not just build and test, but review, promote, changelog, collaborate, etc.
2. Integrate best of breed software in this ecosystem to achieve it
3. Kubernetes as foundation & means to an end
4. Multicloud: Democratize it by building a high level, pleasant CLI
5. GitOps: deployments should be recorded and tracked in Git
THE SCIENCE OF DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations

Nicole Forsgren, PhD
Jez Humble and Gene Kim

Capabilities of Jenkins X

1. Use version control for all artifacts.
2. Automate your deployment process.
3. Use trunk-based development.
4. Implement continuous integration.
5. Kubernetes continuous delivery.
6. Use loosely coupled architecture.
7. Architect for empowered teams.

Jenkins X uses capabilities identified by the Accelerate book by Nicole Forsgren, Jez Humble & Gene Kim

https://jenkins-x.io/about/accelerate

#VelocityConf @tracymiranda
Best Practices for CI/CD for cloud native apps

1. Figure out the best practices of how to CD cloud native apps
   - Not just build and test, but review, promote, changelog, collaborate, etc.

2. Integrate best of breed software in this ecosystem to achieve it

3. Kubernetes as foundation & means to an end

4. Multicloud: Democratize it by building a high level, pleasant CLI

5. GitOps: deployments should be recorded and tracked in Git
Best Practices for CI/CD for cloud native apps

1. Figure out the best practices of how to CD cloud native apps
   • Not just build and test, but review, promote, changelog, collaborate, etc.

2. Integrate best of breed software in this ecosystem to achieve it

3. Kubernetes as foundation & means to an end

4. Multicloud: Democratize it by building a high level, pleasant CLI

5. GitOps: deployments should be recorded and tracked in Git
Kubernetes Custom Resource Definitions (CRD)

Environment CRD

```go
// EnvironmentSpec is the specification of an Environment
type EnvironmentSpec struct {
    Label string `json:"label,omitempty" protobuf:"bytes,1,opt,name=label"`
    Namespace string `json:"namespace,omitempty" protobuf:"bytes,2,opt,name=namespace"`
    Cluster string `json:"cluster,omitempty" protobuf:"bytes,3,opt,name=cluster"`
    PromotionStrategy PromotionStrategyType `json:"promotionStrategy,omitempty" protobuf:"bytes,4,opt,name=promotionStrategy"`
    Source EnvironmentRepository `json:"source,omitempty" protobuf:"bytes,5,opt,name=source"`
    Order int32 `json:"order,omitempty" protobuf:"bytes,6,opt,name=order"`
    Kind EnvironmentKindType `json:"kind,omitempty" protobuf:"bytes,7,opt,name=kind"`
    PullRequestURL string `json:"pullRequestURL,omitempty" protobuf:"bytes,8,opt,name=pullRequestURL"`
    TeamSettings TeamSettings `json:"teamSettings,omitempty" protobuf:"bytes,9,opt,name=teamSettings"`
    PreviewGitSpec PreviewGitSpec `json:"previewGitInfo,omitempty" protobuf:"bytes,10,opt,name=previewGitInfo"`
    WebHookEngine WebHookEngineType `json:"webHookEngine,omitempty" protobuf:"bytes,11,opt,name=webHookEngine"`
}
```
Kubernetes Custom Resource Definitions (CRD)

Pipeline Activity CRD

```go
// PipelineActivitySpec is the specification of the pipeline activity

type PipelineActivitySpec struct {
    Pipeline string `json:"pipeline",omitempty" proto buf:"bytes,1,opt,name=pipeline"
    Build string `json:"build",omitempty" proto buf:"bytes,2,opt,name=build"
    Version string `json:"version",omitempty" proto buf:"bytes,3,opt,name=version"
    Status ActivityStatusType `json:"status",omitempty" proto buf:"bytes,4,opt,name=status"
    StartedTimestamp metav1.Time `json:"startedTimestamp",omitempty" proto buf:"bytes,5,opt,name=startedTimestamp"
    CompletedTimestamp metav1.Time `json:"completedTimestamp",omitempty" proto buf:"bytes,6,opt,name=completedTimestamp"
    Steps []PipelineActivityStep `json:"steps",omitempty" proto buf:"bytes,7,opt,name=steps"
    BuildURL string `json:"buildURL",omitempty" proto buf:"bytes,8,opt,name=buildURL"
    BuildLogsURL string `json:"buildLogsURL",omitempty" proto buf:"bytes,9,opt,name=buildLogsURL"
    GitURL string `json:"gitURL",omitempty" proto buf:"bytes,10,opt,name=gitURL"
    GitRepository string `json:"gitRepository",omitempty" proto buf:"bytes,10,opt,name=gitRepository"
    GitOwner string `json:"gitOwner",omitempty" proto buf:"bytes,10,opt,name=gitOwner"
    ReleaseNotesURL string `json:"releaseNotesURL",omitempty" proto buf:"bytes,11,opt,name=releaseNotesURL"
    LastCommitSHA string `json:"lastCommitSHA",omitempty" proto buf:"bytes,12,opt,name=lastCommitSHA"
    LastCommitMessage string `json:"lastCommitMessage",omitempty" proto buf:"bytes,13,opt,name=lastCommitMessage"
    LastCommitURL string `json:"lastCommitURL",omitempty" proto buf:"bytes,14,opt,name=lastCommitURL"
    Workflow string `json:"workflow",omitempty" proto buf:"bytes,15,opt,name=workflow"
    WorkflowStatus ActivityStatusType `json:"workflowStatus",omitempty" proto buf:"bytes,16,opt,name=workflowStatus"
    WorkflowMessage string `json:"workflowMessage",omitempty" proto buf:"bytes,17,opt,name=workflowMessage"
    PostExtensions []ExtensionExecution `json:"postExtensions",omitempty" proto buf:"bytes,18,opt,name=postExtensions"
    Attachments []Attachment `json:"attachments",omitempty" proto buf:"bytes,19,opt,name=attachments"
    Facts []Fact `json:"facts",omitempty" proto buf:"bytes,20,opt,name=facts"
}
```
Best Practices for CI/CD for cloud native apps

1. Figure out the best practices of how to CD cloud native apps
   • Not just build and test, but review, promote, changelog, collaborate, etc.

2. Integrate best of breed software in this ecosystem to achieve it

3. Kubernetes as foundation & means to an end

4. Multicloud: Democratize it by building a high level, pleasant CLI

5. GitOps: deployments should be recorded and tracked in Git
“Friends don’t let friends install Kubernetes”

- James Strachan, Jenkins X
You can have fun tinkering.
Or you can just get productive
MultiCloud Support

jx create cluster minikube

- aws (Amazon Web Services via kops - https://github.com/aws-samples/aws-workshop-for-kubernetes/blob/master/readme)
- eks (Amazon Web Services Elastic Container Service for Kubernetes - https://docs.aws.amazon.com/eks/latest/userguide)
- gke (Google Container Engine - https://cloud.google.com/kubernetes-engine)
- oke (Oracle Cloud Infrastructure Container Engine for Kubernetes - https://docs.cloud.oracle.com/iaas/Content/Con
- icp (IBM Cloud Private) - https://www.ibm.com/cloud/private
- iks (IBM Cloud Kubernetes Service - https://console.bluemix.net/docs/containers)
- oke (Oracle Cloud Infrastructure Container Engine for Kubernetes - https://docs.cloud.oracle.com/iaas/Content/Con
- kubernetes for custom installations of Kubernetes
- minikube (single-node Kubernetes cluster inside a VM on your laptop)
- minishift (single-node OpenShift cluster inside a VM on your laptop)
- openshift for installing on 3.9.x or later clusters of OpenShift
MultiCloud Create Cluster

- Install kubectl, Helm, Jenkins X, etc

- Install cloud provider CLI
  - az, gcloud, kops, eksctl, oci

- Create Kubernetes cluster
  - RBAC, default storage class, enable docker registries, etc

- Configure
  - Create default namespace
  - Setup the ingress controller
  - Configure git source repository
  - Create admin secrets
Best Practices for CI/CD for cloud native apps

1. Figure out the best practices of how to CD cloud native apps
   - Not just build and test, but review, promote, changelog, collaborate, etc.

2. Integrate best of breed software in this ecosystem to achieve it

3. Kubernetes is a means to an end

4. Multicloud: Democratize it by building a high level, pleasant CLI

5. GitOps: deployments should be recorded and tracked in Git
Best Practices for CI/CD for cloud native apps

1. Figure out the best practices of how to CD cloud native apps
   • Not just build and test, but review, promote, changelog, collaborate, etc.

2. Integrate best of breed software in this ecosystem to achieve it

3. Kubernetes is a means to an end

4. Multicloud: Democratize it by building a high level, pleasant CLI

5. GitOps: deployments should be recorded and tracked in Git
Future CI/CD
“Towards Progressive Delivery”
James Governor, Redmonk

• ‘Blast Radius’
• Feature flags
• Canary
• ‘Built for failure’
Serverless Jenkins with Jenkins X

“By the time you merge it’s too late” – shifting left on CI/CD for pull requests

Headless Jenkins (Jenkinsfile Runner)

Knative Build

Prow

Jenkins X

https://medium.com/@jdrawlings/serverless-jenkins-with-jenkins-x-9134cbfe6870
Wrap-up
Wrap Up

- Cloud native & Kubernetes are a major industry shift
- GitOps has essential techniques and practices to help you go faster without costing you stability.
- Jenkins X focuses on best practices for CI/CD for Kubernetes
- CI/CD is evolving forward quickly: serverless, progressive delivery
- We’re all on this journey, let’s drive things forward together