GitOps и инструменты поставки

Не забудь включить запись!

План

1. Обозначим проблему и посмотрим, как индустрия её решает
2. Познакомимся с концепцией GitOps
3. Обзор k8s-native/gitops-based инструментов поставки
4. Подведем итоги сравнительного обзора

Проблематика

- Что-то "извне" управляет состоянием кластера
  - CI инструмент
  - `kubectl apply`
- Continious Convergency
  - Применить изменения ≠ Измененения применены
  - Связь между кодом к SVC и стейтом кластера неочевидна
  - CI инструменты не могут нормально отследить релиз
- Приятно бы иметь современные модели деплоя

GitOps

We cannot say what actual state is. We can only observe it

GitOps | Intro

Концепция управления конфигурацией кластера

1. Git - signle point of truth для декларативно описанной конфигурации кластера и кода самого приложения
2. В менеджменте кластера больше не участвуют:
    - kubectl не используется напрямую
    - CI не имеет доступа к кластеру
3. k8s controller для управления конфигурацией кластера.
4. Используется *git pull* модель доставки изменений
5. Immutable containers
6. Git Pull модель применения изменений

GitOps | Immutable Containers

- Контейнер не подвергается никаким изменениям на этапе эксплуатации:
  - никаких патчей, обновлений во время эксплуатации
  - один контейнер используется на всех окружениях
    - параметризуется с помощью *Secrets*, *ConfigMaps*

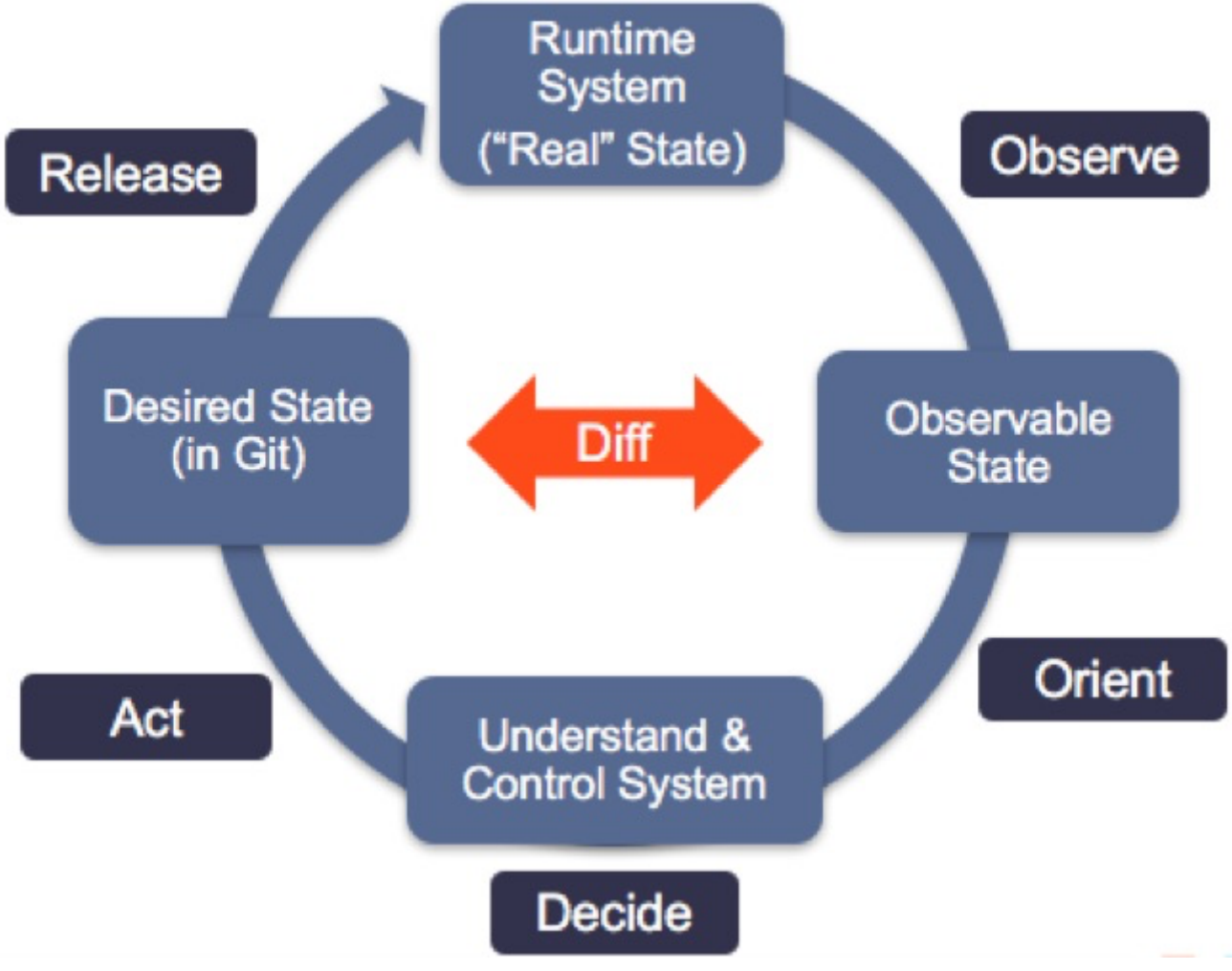
GitOps | Pull model

1. Ни один внешний клиент не имеет прав на внесение изменений в кластер, все обновления накатываются изнутри.
2. Docker Registry сканируется на наличие новых версий. Если появляется новый образ, Git-репозиторий и deployment обновляются на новую версию.
3. Pull-инструменты могут быть распределены по разным пространствам имен с разными репозиториями Git и правами доступа.
4. Отсутствует связь с CD-пайплайнами, поскольку развертывания происходят внутри кластера. Cluster credentials остаются лишь внутри кластера.

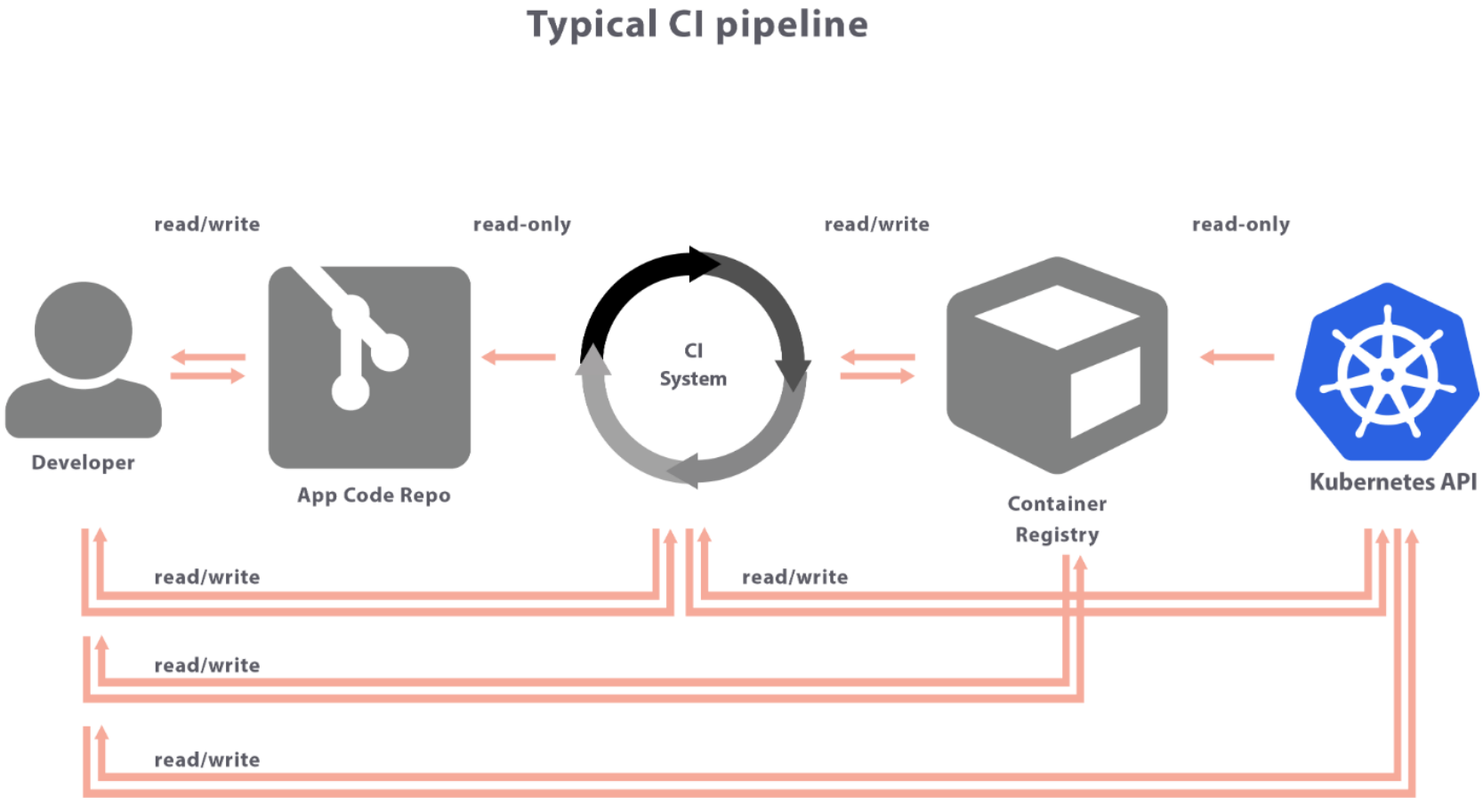
GitOps | Convergency

Механизм применения изменений конфигурации кластера оркестратором, не прекращающийся до тех пор, пока конфигурация кластера не станет идентичной описанной в репозитории.

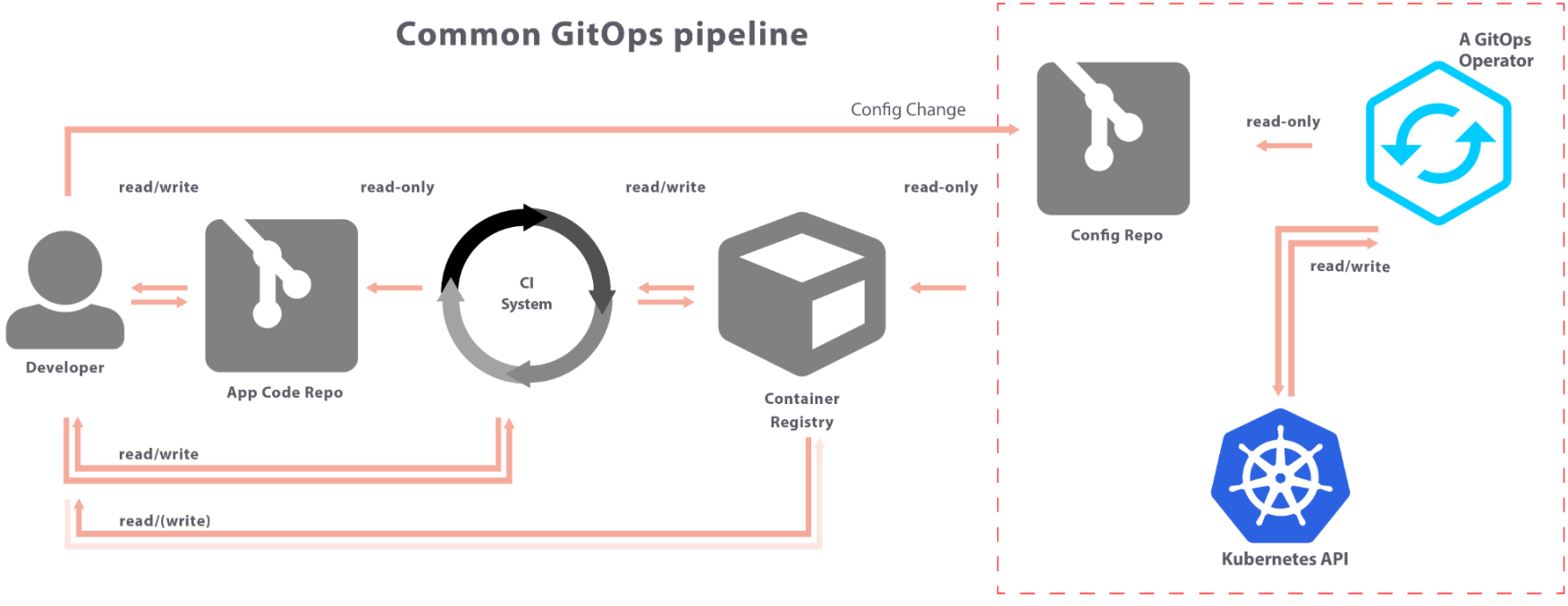
GitOps | ODAA-loop extension



GitOps | CI pipeline



GitOps | GitOps pipeline

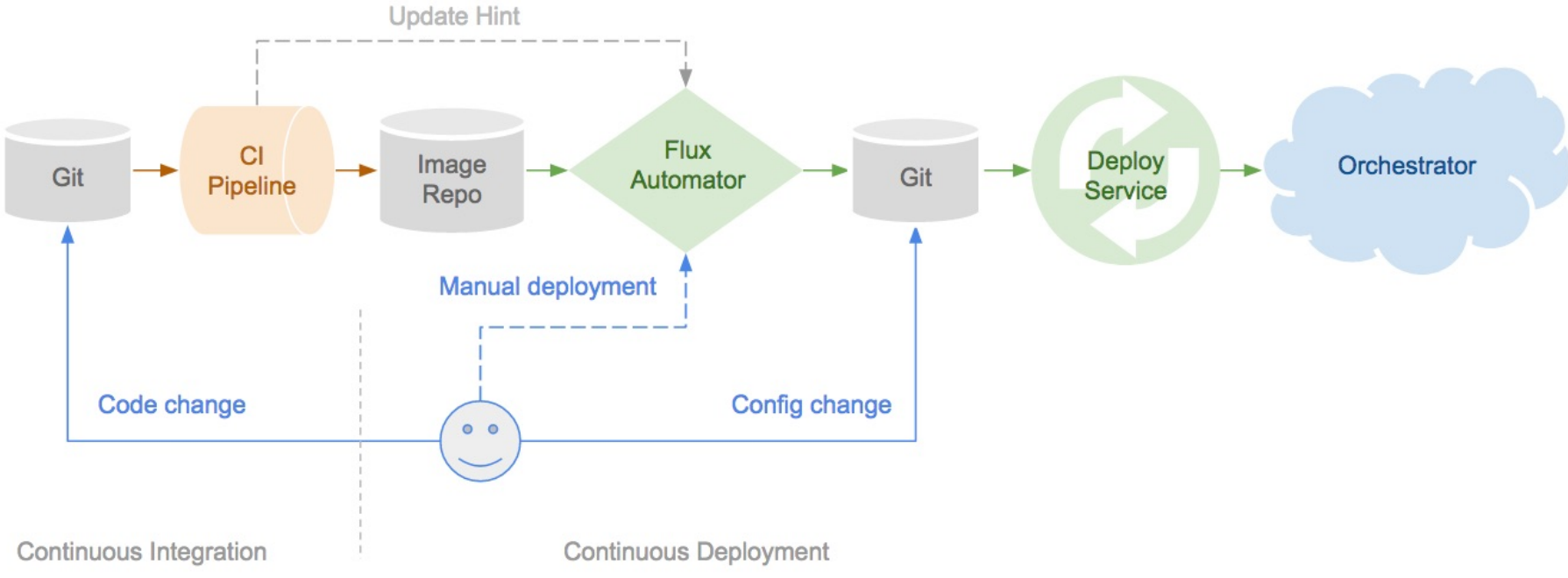


GitOps | Typical workflow

- Feature-workflow разработчика компании WeaveWorks:

  1. Инженер отправляет *pull request* из feature-ветки
  2. Code Reivew, PR approve
  3. Git hooks тригерят CI-пайплайн: тестирование изменений и выгрузка успешного билда в registry
  4. *Deployment Automator* отслеживает появление новой версии образа приложения в registry и обновляет репозиторий с конфигурацией кластера
  5. *Deployment Synchronizer* синхронизирует текущее состояние кластера с описанным в конфигурационном репозитории

GitOps | Typical workflow



GitOps | How can I get it?

- VCS + IaC
- Container Registry для CI
- Controller with 2-way sync

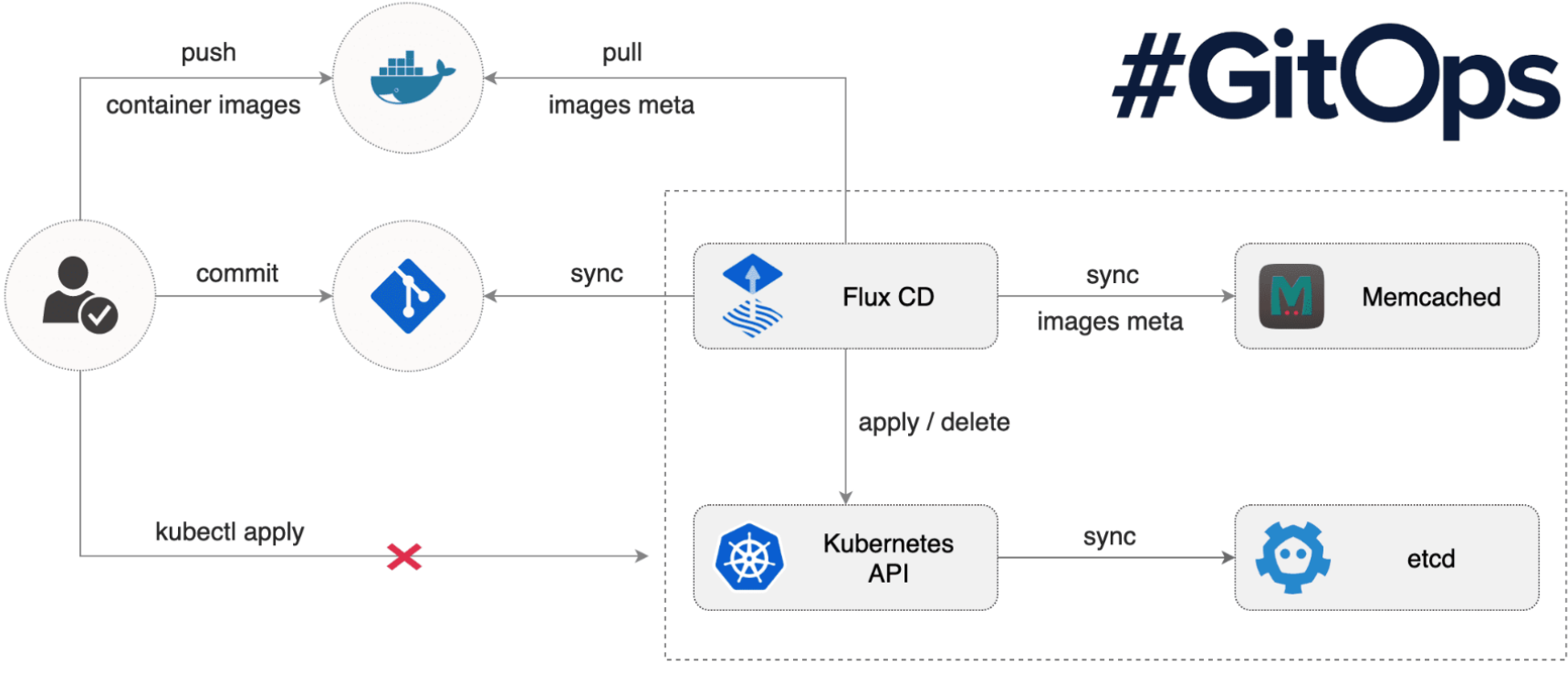
Flux

Flux | Intro

- Flux - k8s Continuous Delivery инструмент от родоначальников GitOps.
  - Автоматизированная синхронизация конфигурации кластера с git-репо (convergency)
  - Автоматический деплой новый образов контейнеров
  - Интеграция с шаблонизаторами (Kustomize, Helm) и Flagger
- fluxCD/helm-operator

> 2983 github stars

Flux | architecture



Flux | Interaction with repo

1. Один репозиторий с манифестами на 1 оператор
2. По умолчанию, просит RW доступ до репо
    - Но может работать и с RO (`--git-readonly`)
    - helm-operator хранит в директории `/releases/` репо свои манифесты типа *HelmRelease*
3. Игнорирует JSON файлы
    - Можно указать, что именно процессить (`--git-path`)
    - Можно указать конкретные ветки репо (`--git-branch`)

...

Flux | Interaction witn repo

...

4. Умеет удалять ресурсы из кластера (`--sync-garbage-collection`)
    - Выключено по умолчанию
5. Имеет настраиваемый интервал синхронизации (5min по умолчанию)
    - `--git-poll-interval` - часто обновления стейта репозитория
    - `--sync-interval` - частота применения обновления к кластеру

Flux | HelmRelease example

```yaml
apiVersion: helm.fluxcd.io/v1
kind: HelmRelease
metadata:
  name: rabbit
  namespace: default
  annotations:
    flux.weave.works/automated: "true"
    flux.weave.works/tag.chart-image: glob:dev-*
spec:
  releaseName: rabbitmq
  targetNamespace: mq
  chart:
    repository: https://kubernetes-charts.storage.googleapis.com/
    name: rabbitmq
    version: 3.3.6
  values:
    replicas: 1
```

Flux | Interaction with images registry

1. Flux наблюдает за registry на основании k8s *ImagePullSecrets*
    - Имеет ограничения и workaround'ы к ним
2. Частота сканирования registry конфигугируется, но не рекомендуется к изменению (флаги `--registry-rps` и `--registry-burst`)
3. Частота применения найденных при сканировании registry изменений также конфигурируема (`--automation-interval`)
4. Можно исключить образы из пула сканирования (`registry-exclude-image`)
    - `--registry-exclude-image=docker.io/*,quay.io/*`

Flux | Interaction with k8s

- Права могут быть ограничены namespace'ом (эксперементальная фича)
  - `--k8s-allow-namespace`
- Может деплоить в разные namespace'ы
- Заявлена совместимость с HPA (через удаление `spec.replicas` из манифеста)
- Можно добавлять ресурсы в *IngoredList* через аннотации:

```
...
  fluxcd.io/ignore: true
...
```

Flux | Integration with Templating tools

- Эксперементальный функционал
  - Файлы расширения `*.flux.yaml`
  - Для включения указать флаг `--manifest-generation=true`
  - Сложно описанная интеграция с Kustomize и helm (репо с примерами недоступны)
- helm-operator
  - helm3 in development

Skaffold

Skaffold | Intro

- Инструмент локальной / удаленной разработки контейниризированных приложений для последующего деплоя в k8s.
- Ключевые фичи
  - Поддерживает рендеринг не только helm тимплейтов
  - Имеет **два режима**: локальной разработки(сборка и деплой) и запуска CICD пайплайна для деплоя в кластер
  - Сборка только директорий с изменениями при хранении нескольких микросервисов в одном trunk'e

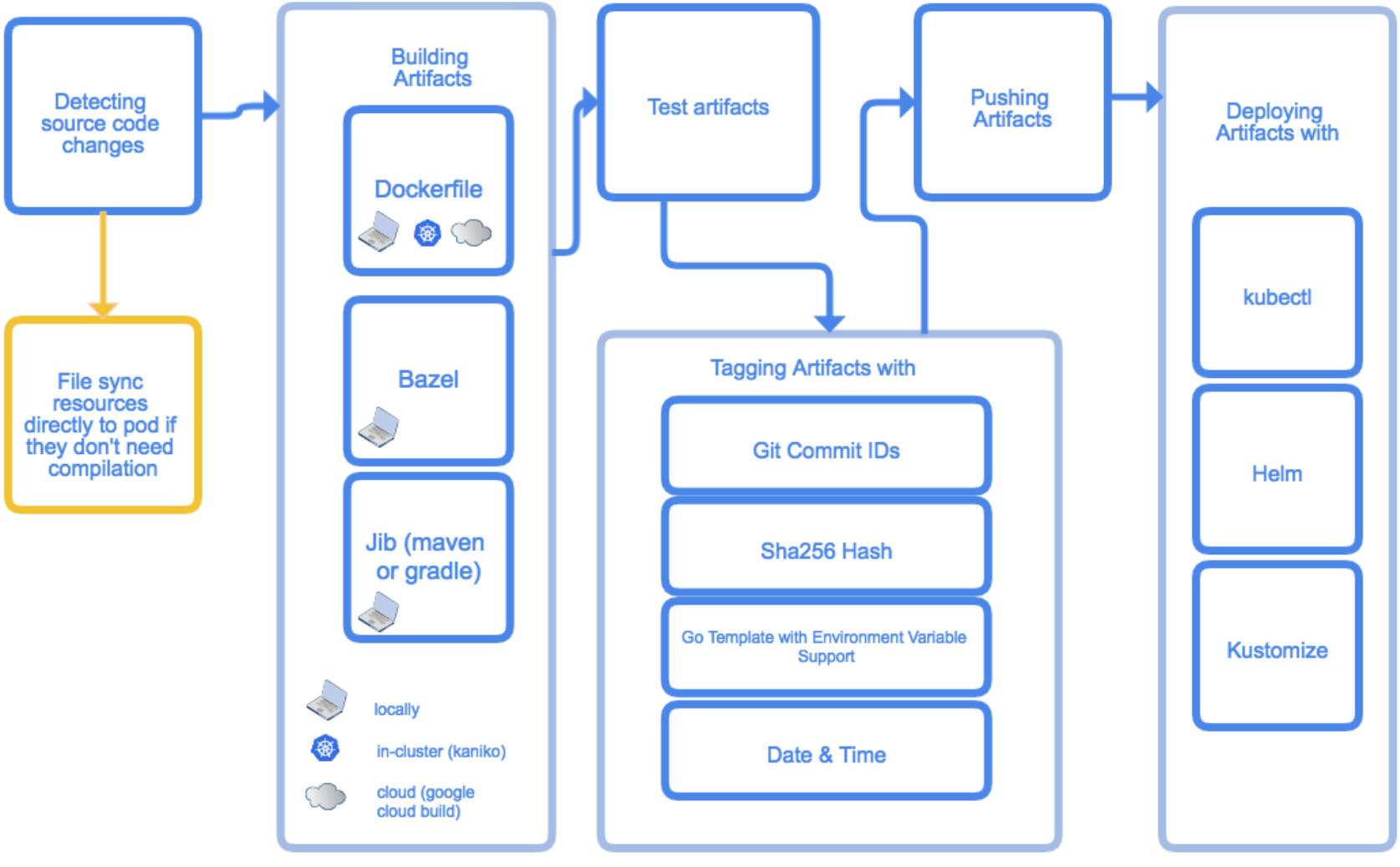
7677 github stars

Skaffold | Workflow - dev

> *Skaffold primarily simplifies the "build → deploy → refactor → repeat" cycle*

1. Сборка образа: локально или удаленно
2. Push в хранилище образов (опционально)
3. Обновление тегов образов в манифестах k8s
4. Деплой с помощью `kubectl apply` или `helm upgrade`
5. Стриминг логов с запущенных инстансов
6. Мониторинг изменений исходников или конфигурационных манфестов в репо для повторения шагов 1-5
7. Удаляет все артефакты за собой по кончанию работы

Skaffold | Workflow



Skaffold | Commands

```
End-to-end pipelines:
  run                 Run a pipeline
  dev                 Run a pipeline in development mode
  debug               Run a pipeline in debug mode

Pipeline building blocks for CI/CD:
  build               Build the artifacts
  deploy              Deploy pre-built artifacts
  delete              Delete the deployed application
  render              Perform all image builds, and output rendered kubernetes manifes

Getting started with a new project:
  init                Generate configuration for deploying an application
  fix                 Update old configuration to newest schema version

Other Commands:
  completion          Output shell completion for the given shell (bash or zsh)
  config              Interact with the Skaffold configuration
  diagnose            Run a diagnostic on Skaffold
  version             Print the version information
```

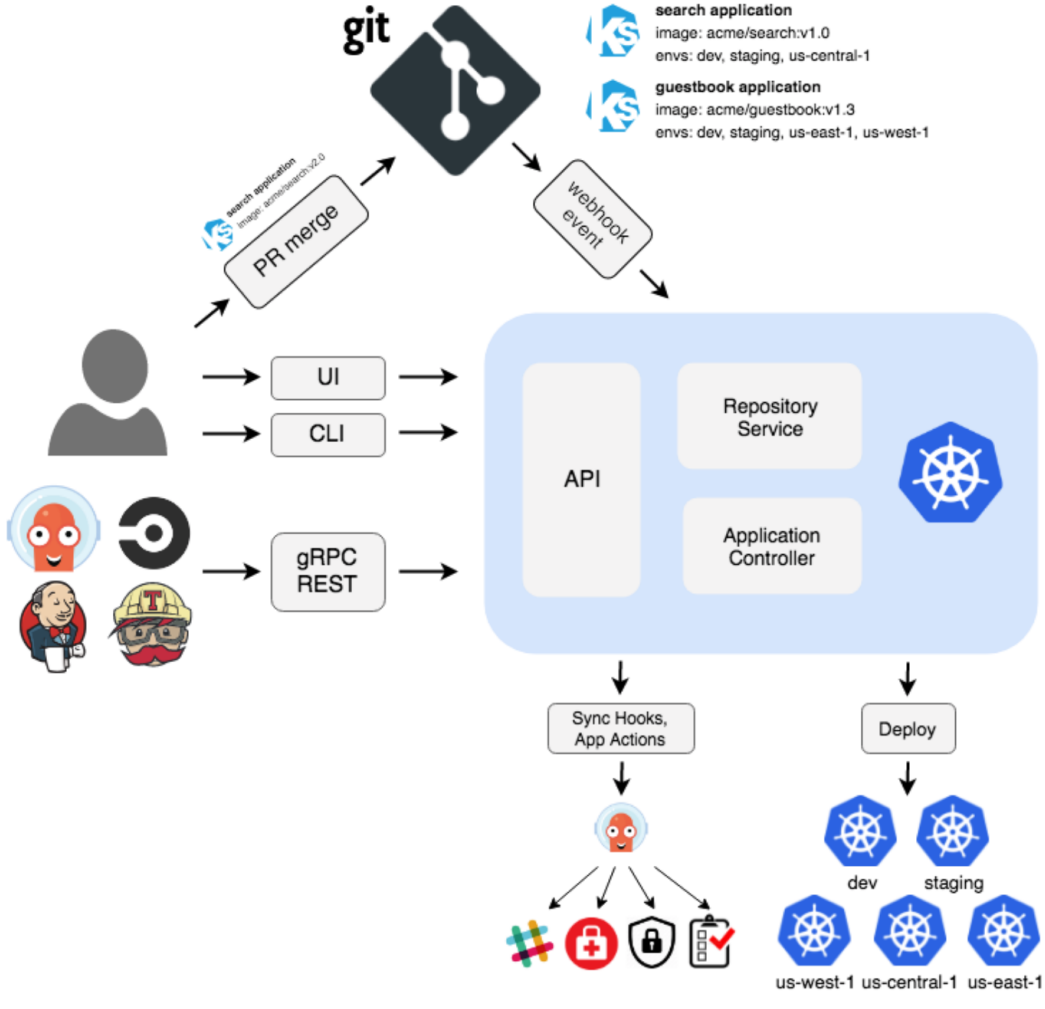
Argo CD

Argo CD | Intro

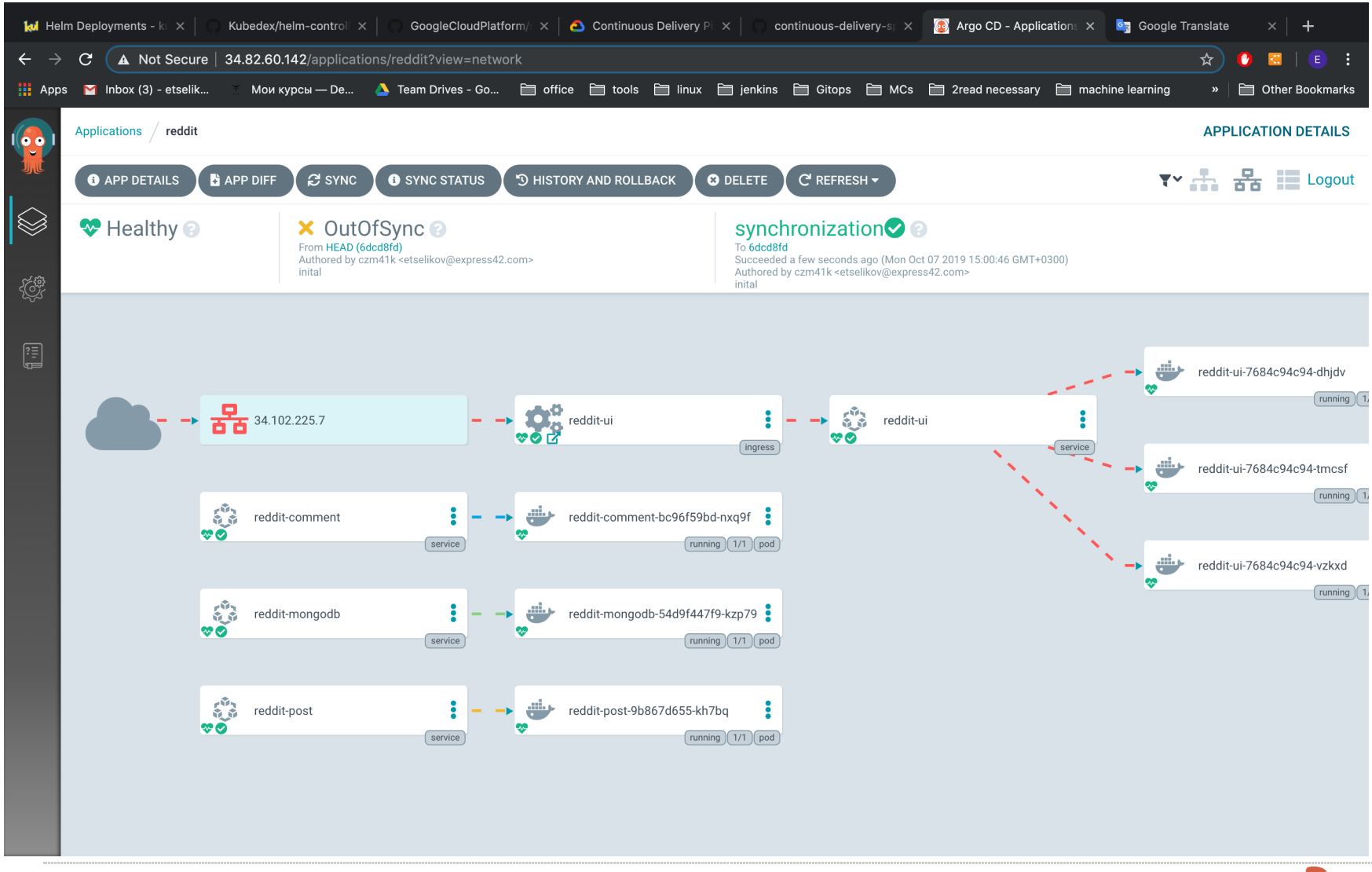
- Декларативный CD-инструмент для реализации GitOps
  - Поддержка шаблонизаторов
    - Свои шаблонизаторы в виде плагинов
  - Rollback/Roll-anywhere
  - UI
    - Health status приложения
  - Prometheus метрики
  - Переключение между кластерами

1769 github stars

Argo CD | Architecture



ArgoCD | UI


Helm Deployments - k
Kubedex/helm-control
GoogleCloudPlatform/
Continuous Delivery P
continuous-delivery-s
Argo CD - Applications
Google Translate
Not Secure
34.82.60.142/applications/reddit?view=network
Apps
Inbox (3) - etselik...
Мои курсы — De...
Team Drives - Go...
office
tools
linux
jenkins
Gitops
MCs
2read necessary
machine learning
Other Bookmarks
Applications
reddit
APPLICATION DETAILS
APP DETAILS
APP DIFF
SYNC
SYNC STATUS
HISTORY AND ROLLBACK
DELETE
REFRESH
Logout
Healthy
OutOfSync
From HEAD (6dcd8fd)
Authored by czm41k <etselikov@express42.com>
inital
synchronization
To 6dcd8fd
Succeeded a few seconds ago (Mon Oct 07 2019 15:00:46 GMT+0300)
Authored by czm41k <etselikov@express42.com>
inital
34.102.225.7
reddit-ui
ingress
reddit-ui
service
reddit-ui-7684c94c94-dhjdv
reddit-ui-7684c94c94-tmcsf
reddit-ui-7684c94c94-vzkxd
running
reddit-comment
service
reddit-comment-bc96f59bd-nxq9f
running
1/1
pod
reddit-mongodb
service
reddit-mongodb-54d9f447f9-kzp79
running
1/1
pod
reddit-post
service
reddit-post-9b867d655-kh7bq
running
1/1
pod

Argo CD | Summary

- ссылка на roadmap
- Отлично справляется с заявленным функционалом
  - Низкий порог входа
  - Прозрачность происходящего
- Неплохая документация с примерами
- Нет CI, интеграций с Registry, поддержки моделей деплоя из коробки

Argo CD | Demo

Argo Rollouts

Argo Rollouts | Intro

Инструмент, реализующий современные модели деплоя в kubernetes

- In-cluster контроллер
- Поддержка как blue/green, так и canary релизов
- Поддержка шаблонизаторов
- Поддержка HPA

> ±200 github stars

Argo Rollouts | Deployment strategies

1. Blue/green
2. Cannary

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: example-rollout
spec:
  replicas: 10
  selector:
    matchLabels:
      app: nginx
  # <RS definition has been cut off>
  strategy:
    canary: #Indicates that the rollout should use the Canary strategy
      maxSurge: "25%"
      maxUnavailable: 0,
      steps:
      - setWeight: 10
      - pause:
          duration: 3600 # 1 hour
      - setWeight: 20
      - pause: {}
```

Argo Rollouts | Templating

- Только Kustomize
  - Зато можно его научить понимать наши CRD с помощью transformer-configs

Prow

Prow | Intro

- Инструмент для расширения автоматизации работы с репозиториями на github
  - Много Use-Cases в разных технологических компаниях
  - Синтаксис завернут в Makefile
- Функционал:
  - Концепция Jobs для *обработки* изменений в git
  - Автоматизация merge request'ов
  - Декларативное описание себя (с Continuous Deployment)
  - UI
  - Prometheus метрики

> 1618 github stars

Prow | Components

1. hook
2. plank - job execution & lifecycle контроллер
3. deck - UI
4. horologium - аналог CRON
5. sinker - сборщик мусора
6. tide - обработка PullRequests

    - I JUST WANT MY PR TO MERGE!

Prow | Concepts

- Prow Jobs
  - Job Images
- Prow Plugins

Prow | Plugins

1. Добавляем плагин:
    - Добавляем в `plugins.yaml`
    - Применяем изменения
        - `make update-plugins`
        - Автоматически при добвленном `update-config` плагине
2. Help
    - README.md
    - `PluginHelp`
3. External плагины
4. Marketplace плагинов

Prow | Jobs

- Pre-submit
- Post-submit
- Periodic

```
pre-sets
```

Prow | Jobs | pre-sets

```
presets:
- labels:                  # a job with these labels/values will have the preset applied
    preset-foo-bar: "true" #   key:value pair must be unique among presets
  env:                     # list of valid Kubernetes EnvVars
  - name: FOO
    value: BAR
  volumes:                 # list of valid Kubernetes Volumes
  - name: foo
    emptyDir: {}
  - name: bar
    secret:
      secretName: bar
  volumeMounts:            # list of valid Kubernetes VolumeMounts
  - name: foo
    mountPath: /etc/foo
  - name: bar
    mountPath: /etc/bar
    readOnly: true
- env:                     # a preset with no labels is applied to all jobs
  - name: BAZ
    value: qux
  volumes:
    # etc...
  volumeMounts:
    # etc...
```

Prow | Job Triggers

- unconditionally and automatically
  - `always_run: true`
- conditionally but automatically
  - `run_if_changed: <some_parameter>`
- conditionally but not automatically
  - `always_run: false` && !`run_if_changed`
- PR comment

Prow | Job Example

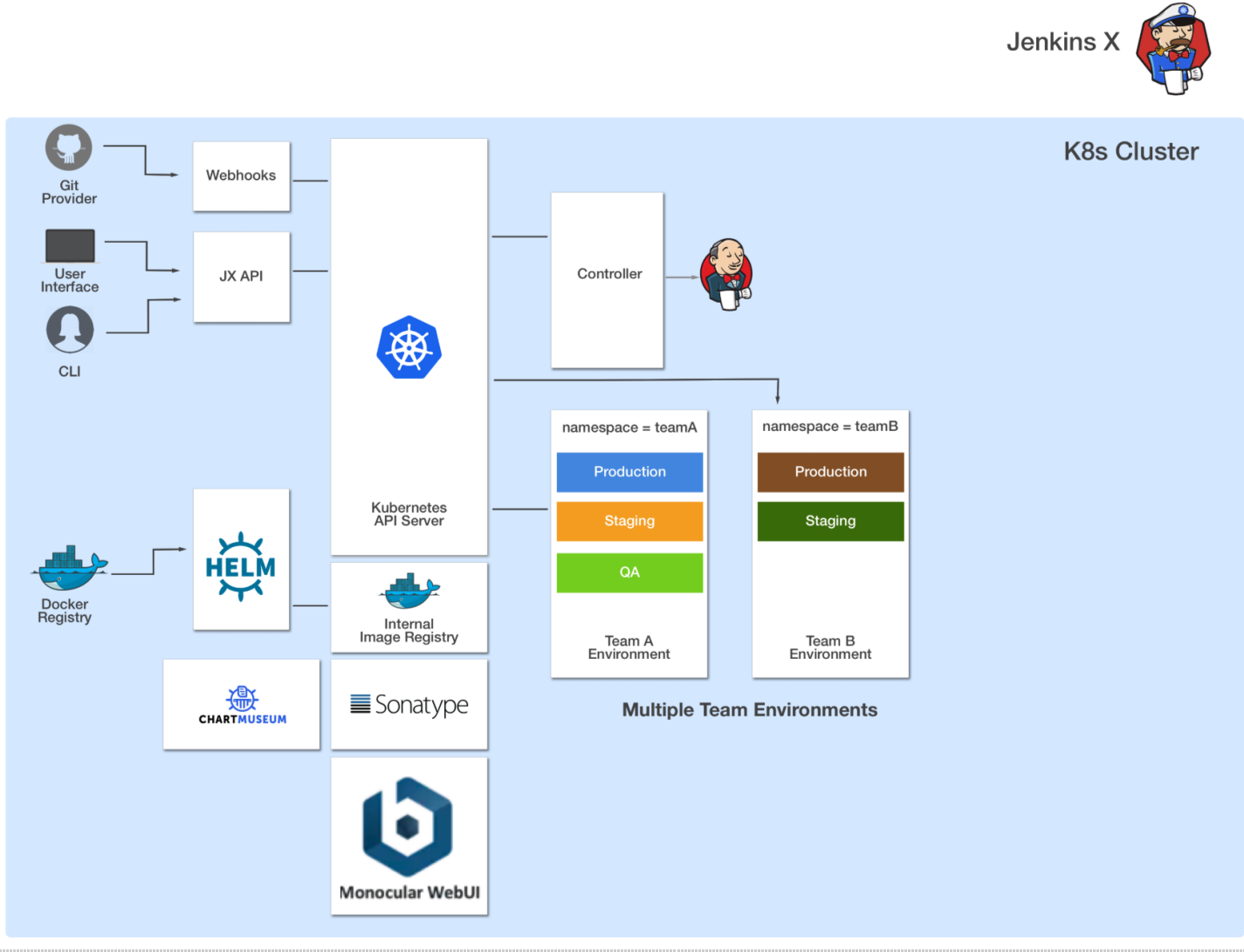
```yaml
presubmits:
  kubernetes/community:
  - name: pull-community-verify  # convention: (job type)-(repo name)-(suite name)
    annotations:
      testgrid-dashboards: sig-contribex-community
      testgrid-tab-name: pull-verify
    branches:
    - master
    decorate: true
    always_run: true
    spec:
      containers:
      - image: golang:1.12.5
        command:
        - /bin/bash
        args:
        - -c
        # Add GOPATH/bin back to PATH to workaround #9469
        - "export PATH=$GOPATH/bin:$PATH && make verify"
```

Prow | Triggering Jobs with Comments

- `/test job-name`
- `/retest`
- `/test all`

JenkinsX

JenkinsX | Architecture



JenkinsX | Features

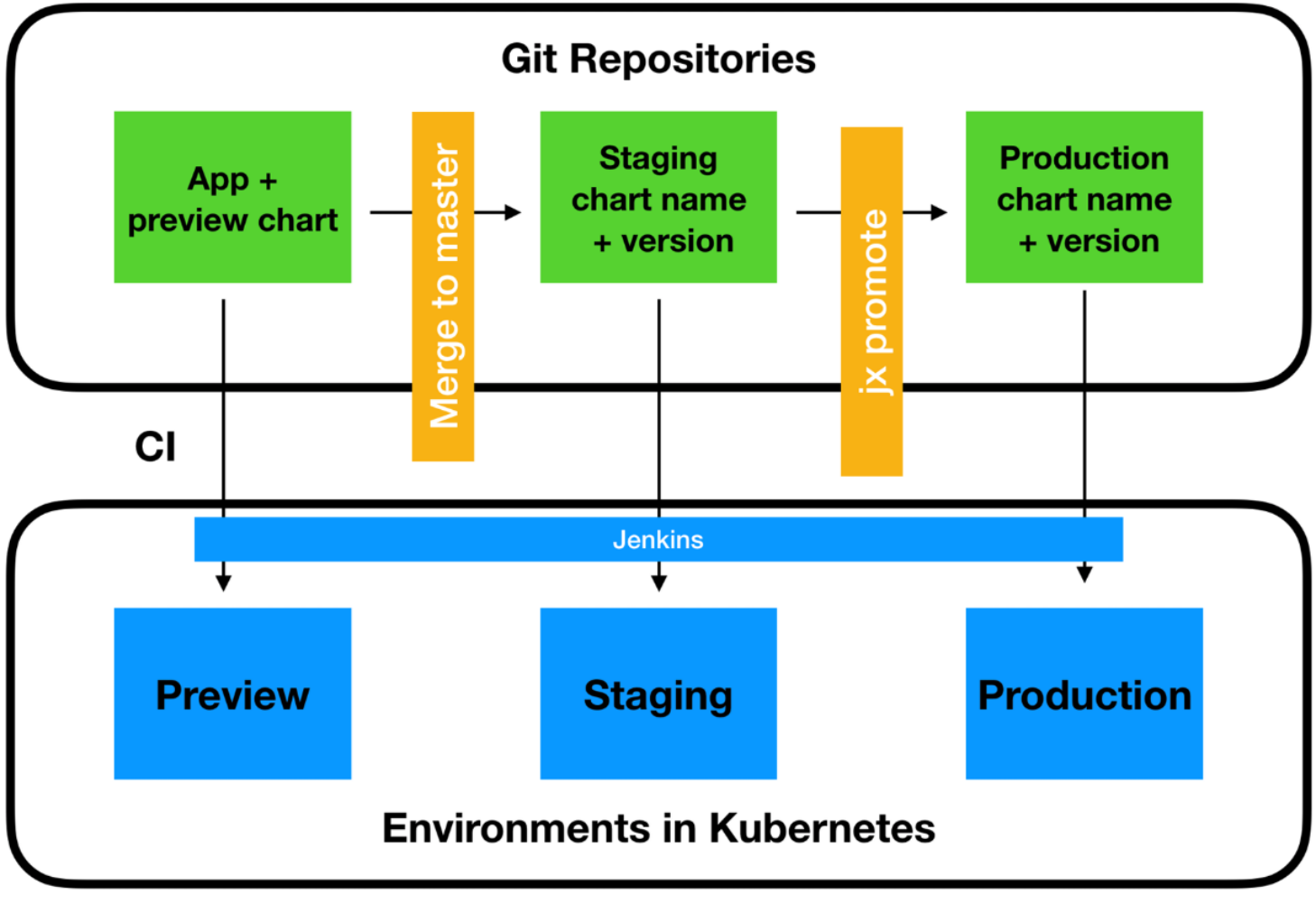
In-cluster инструмент-аггрегатор, предоставляющий почти все инструменты разработки прямо изнутри k8s

1. Quickstart / import
2. Static Jenkins / Jenkins-X Pipelines режимы использования
3. Progressive Delivery possibilities
4. Декларативное описание
5. `jx promote` промоушн
6. Поддержка шаблонизатора helm
7. Preview окружение
8. DevPods

JenkinsX | Environements

- Prod / Staging (default set)
- Preview
- Dev
  - Nexus
  - Jenkins
  - Docker Registry
  - Chart Museum
  - Monocular

JenkinsX | Workflow



JenkinsX | Components

- CRDs
- Skaffold
- Prow
- Lighthouse
- Docker Registry
- Helm Chart Museum
- Tekton Pipelines

JenkinsX | buildpacks

- jenkins-x-kubernetes official builpack
- Описываются дефолтные шаги сборки того или иного стека
- Композитный `pipeline.yml`:
  - release pipeline
  - PR pipeline
  - Feature pipeline

JenkinsX | Personal experience

1. Много очень разных мануалов по его запуску в очень разных конфигурациях
2. Вроде умеет сам себя устанавливтаь, но от этого только сложнее
3. Мало документации и она разная
4. Выглядит крайне перспективно, особенно концепция DevPods
5. Куча багов в jx promote

Spinnaker

Spinnaker | Intro

1. Open-source multi-cloud CD-платформа
    - Не совсем о GitOps
    - Без CI
    - Декларативное описание собственной конфигурации
    - Armory - "Production Graded Spinnaker"
2. 2 заявленных функциональных области:
    - application management
    - CD pipelines

Spinnaker | Features

1. UI
2. Pipelines-As-Code
3. Multi-providers
4. *Modern Rollout strategies(?!)*
5. armory/spinnaker-operator - early release
6. Helm charts

Spinnaker | Extra Info

- Armory/spinnaker-operator - early release
- Инструмент компании Netflix

Spinnaker | Components

1. Orca
2. Deck
3. Gate
4. Clouddriver
5. Front50
6. Igor
7. Halyard

...

Spinnaker | Components-2

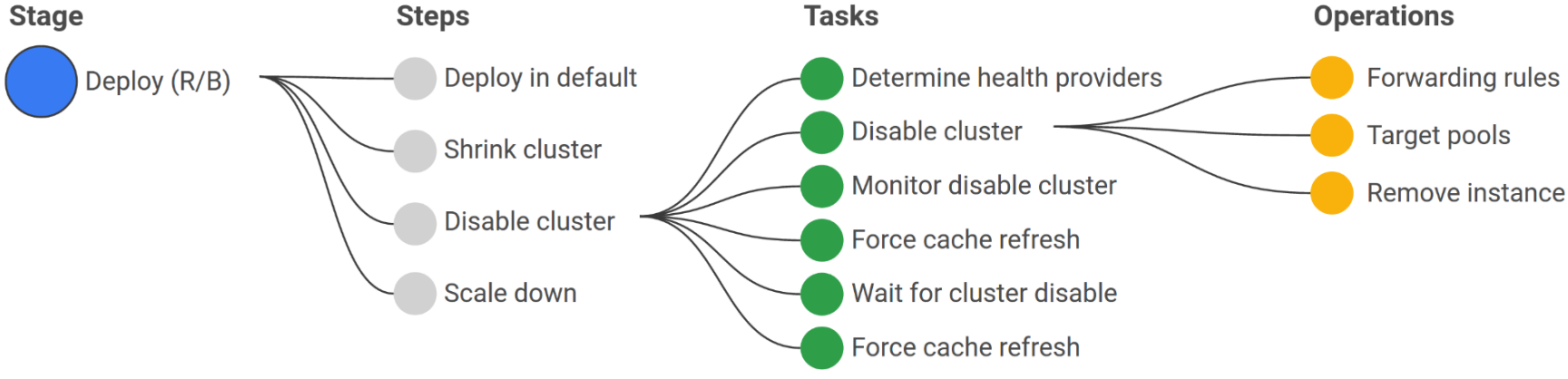
...

8. Kayenta
9. Fiat
10. Echo
11. Rosco

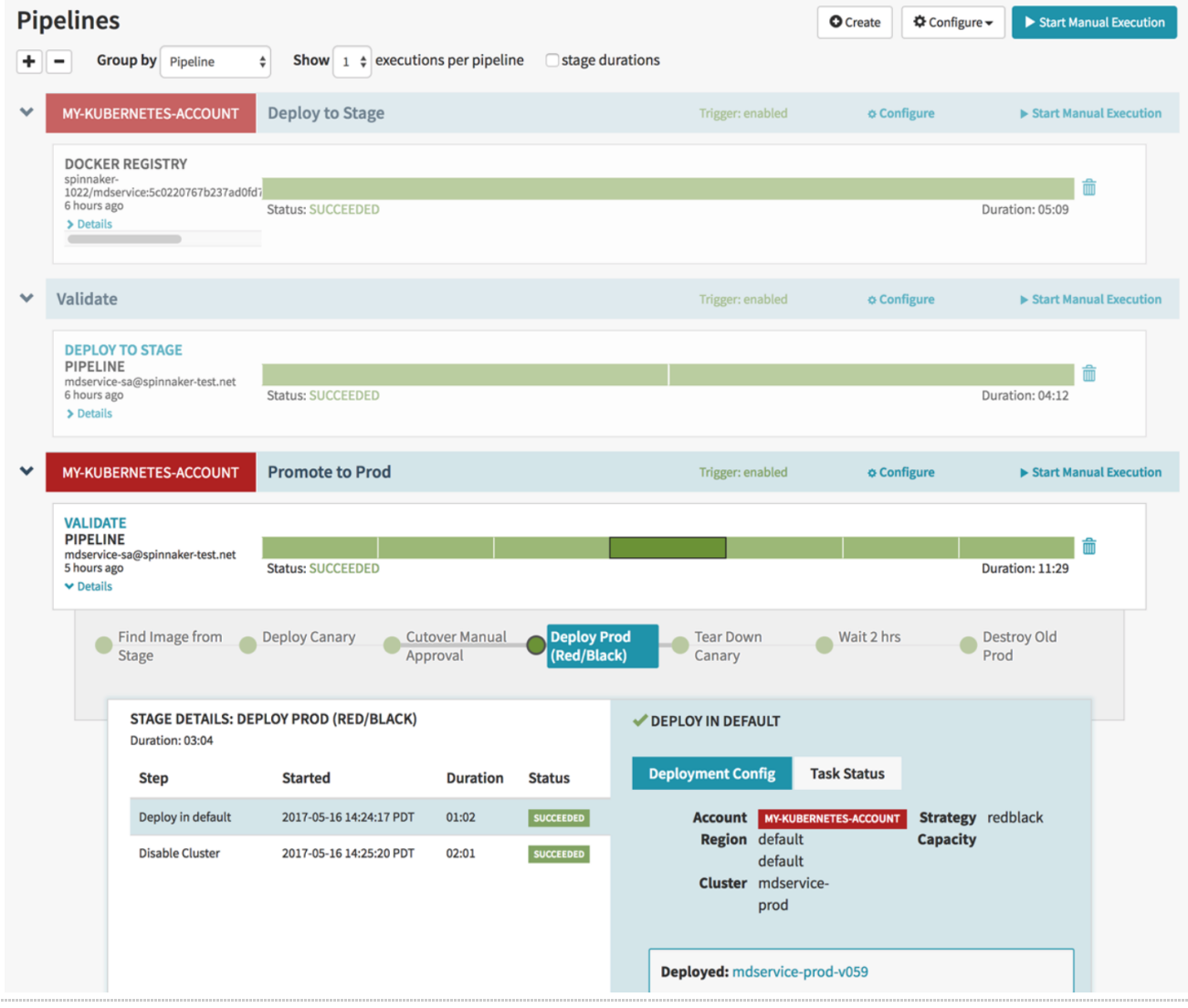
Spinnaker | Pipelines

- *Pipeline=Pipeline template + configuration*
- Delivery modes:
  - Red/Black (aka Blue/Green)
  - Canary Analisys
    - Canary Judje

Spinnaker | Pipelines example



Spinnaker | UI example



Spinnaker | Cannary Analisys

- Canary feature enabled
- Canary configuration
  - multiple canaries per app
- Canary analisys stage
  - RealTime
  - Retrospective
- Canary Judje

> "This stage type is for the canary analysis only. The canary stage doesn’t perform any provisioning or cleanup operations for you. Those must be configured elsewhere in your pipeline." (c) Documentation

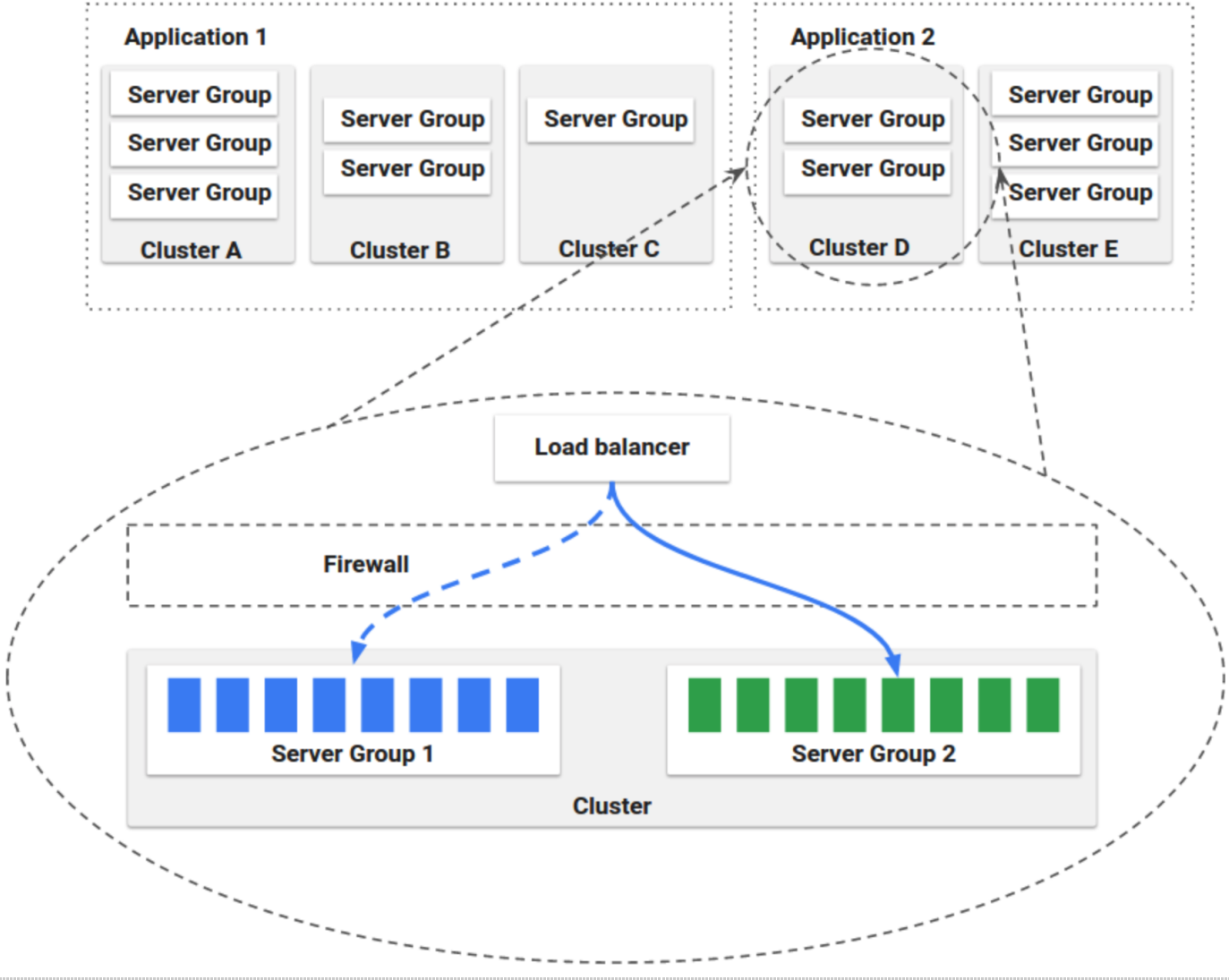
Spinnaker | Cannary Judje

- Процесс принятия решения о стабильности канареечной выкладки.
- Алгоритм (по каждой метрике)

    1. Data validation
    2. Data cleaning
    3. Metrics comparison
    4. Score computation

Default Judje NetflixACAJudge

Spinnaker | Application Management



Spinnaker | Application Management

1. пайплайны
2. канареечные конфиги
3. инфраструктура
    - clusters
    - firewalls
    - load balancers
    - server groups

Spinnaker | Application Management-2

1. Server Groups ≃ Workloads
2. Load Balancers ≃ Services, Ingresses
3. Firewalls ≃ NetworkPolicies
4. Cluster ≃ Deployment (optional)

Spinnaker | Personal Experience

1. Сложен в установке
2. Высокий порог входа
3. Неочевидный синтаксис пайплайнов
4. Неочевидный траблшутинг
5. Куча открытых Issues на github

Не одним spinnaker'ом...

- Flagger
- Argo Rollouts

Werf

Werf | Intro

Open-source k8s-native CI/CD инструмент от компании flant

1280 github stars

Werf | Intro

- Локальная разработка и сборка
  - сборщик Stapel - альтернативный способ сборки
- Оператор ковергенции (Tracker)
- Интеграция в другие CI/CD инструменты
  - `werf ci-env`
- Поддержка шаблонизаторов
- 3-way-merge (in development stage)

Werf | werf.yaml

- Минимальная конфигурация

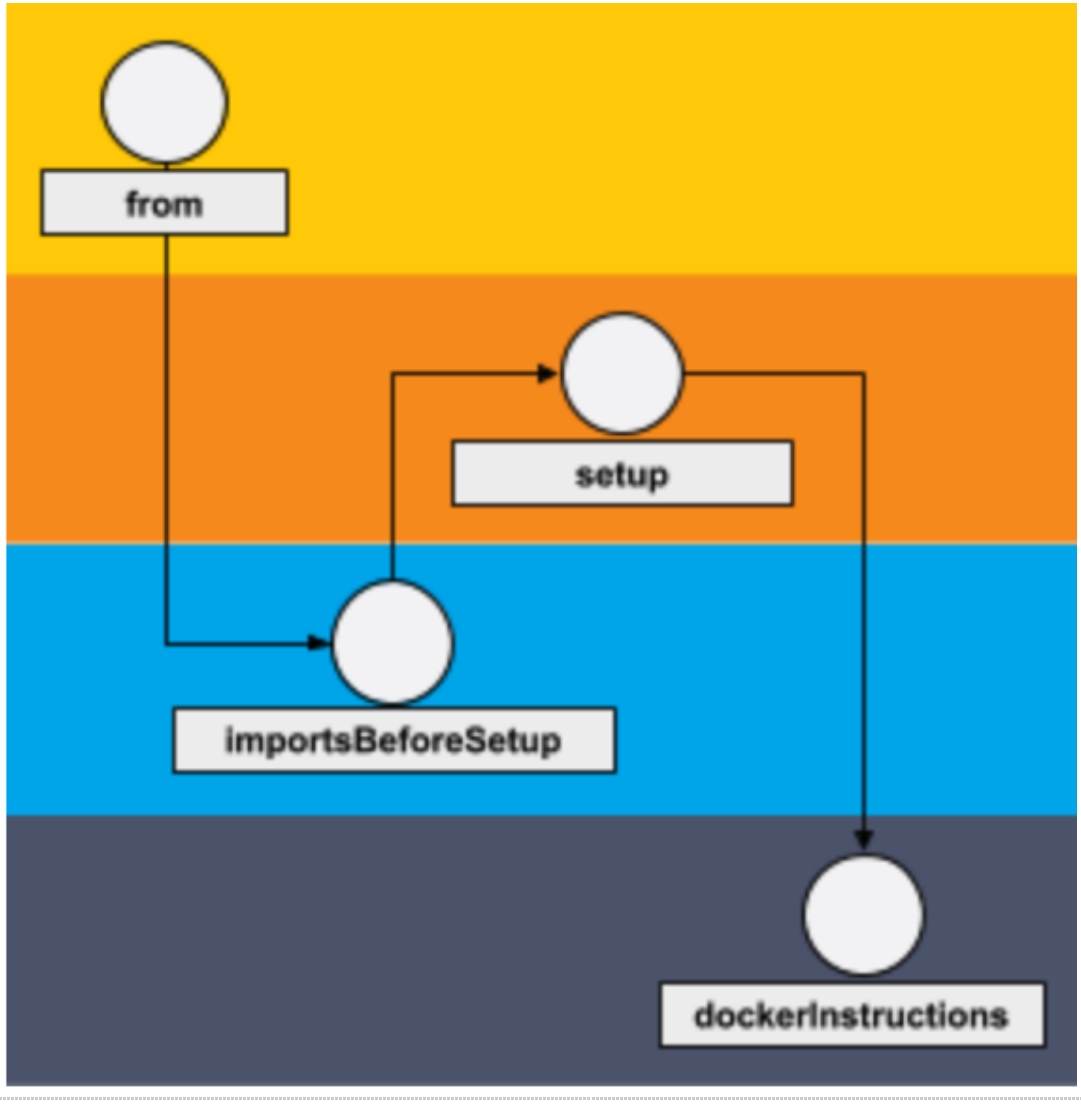
```
project: my-project
configVersion: 1
---
image: ~
from: alpine:latest
```

- Дробление конфигурации (`/.werf/`)

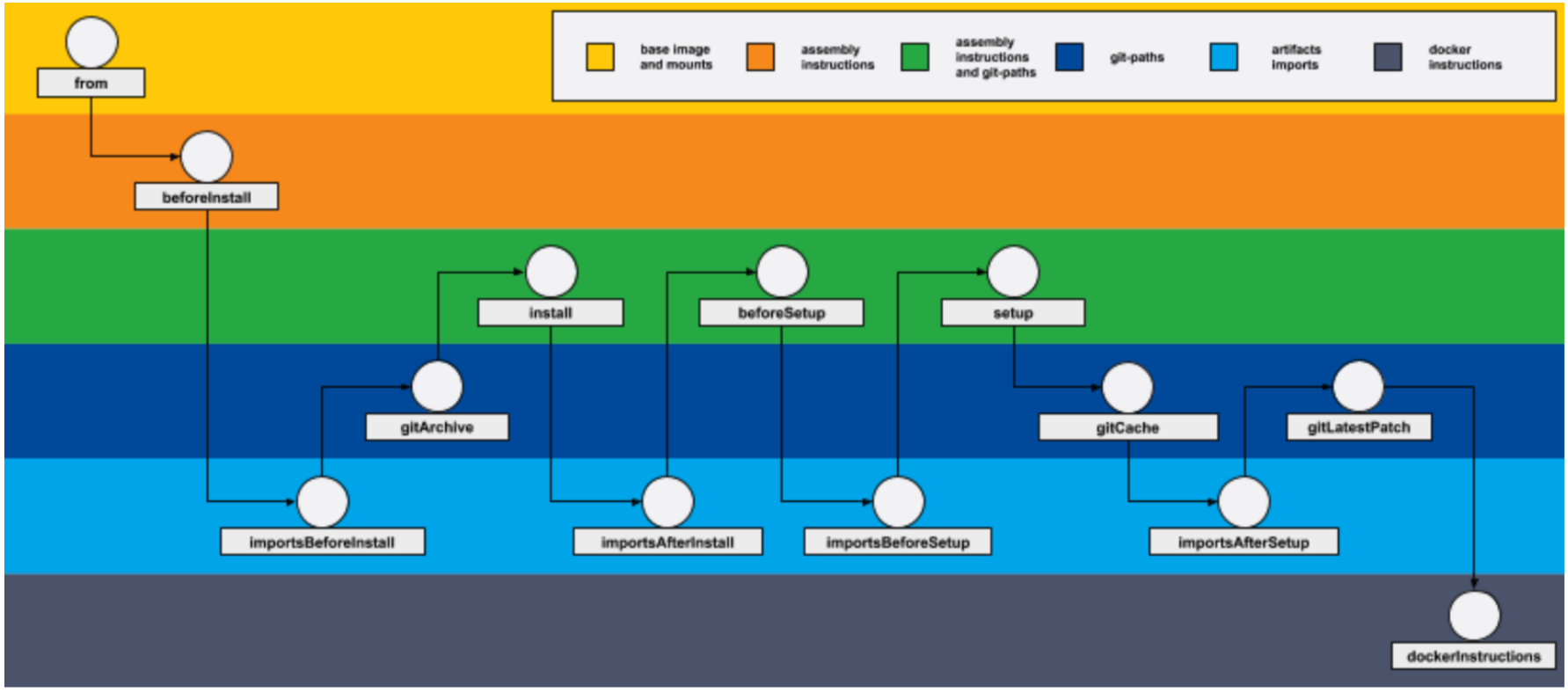
Werf | Build process

- Regular build
  - `--stages-storage :local`
- Stapel build
  - stages: *beforeInstall*, *Install*, *beforeSetup*, *Setup*
  - integration with git and incremental rebuilds
  - allows Ansible
  - cache sharing
  - reduces image size by detaching source data and build tools

Werf | Docker build



Werf | Stapel build



Werf | Publish - Naming

1. Project = 1 single nameless images (`image: ~`)
    - *IMAGES_REPO:TAG*
2. Project = multiple images **from single repo**
    - *IMAGES_REPO:IMAGE_NAME-TAG*
3. Project = multiple images **from different repos**
    - *IMAGES_REPO/IMAGE_NAME:TAG*

```
--images-repo / --images-repo-mode
```

Werf | Publish - Tagging

- --tag-* param
  - `--tag-git-tag`
  - `--tag-git-branch`
  - `--tag-git-commit`
  - `--tag-custom`

Werf | Templates | helm

- values.yaml опционален
  - секция `global`
  - service values
- расположен в `.helm` директории
- интеграция с свежесобранными образами
  - werf_container_image
  - werf_container_env
  - imagePullPolicy параметр

Werf | User-defined values

- `/.helm/values.yaml`(optionaL)

```
global:
  names:
  - alpha
  - beta
  - gamma
  mysql:
    staging:
      user: mysql-staging
    production:
      user: mysql-production
    _default:
      user: mysql-dev
      password: mysql-dev
```

Werf | Service Values

```
global:
  env: stage
  namespace: myapp-stage
  werf:
    ci:
      branch: mybranch
      is_branch: true
      is_tag: false
      ref: mybranch
      tag: '"-"'
    docker_tag: mybranch
    image:
      assets:
        docker_image: registry.domain.com/apps/myapp/assets:mybranch
        docker_image_id:
sha256:ddaec322ee2c622aa0591177062a81009d9e52785be6915c5a37e822c2019755
      rails:
        docker_image: registry.domain.com/apps/myapp/rails:mybranch
        docker_image_id:
sha256:646c56c828beaf26e67e84a46bcdb6ab555c6bce8ebeb066b79a9075d0e87f50
    is_nameless_image: false
    name: myapp
    repo: registry.domain.com/apps/myapp
```

Werf | Deploy process

- Releases (release-names)
- Environments
- `werf deploy`
  - render -> run pre-install pre-upgrade helm hooks
  - apply changes
  - create new version and save current resources manifests state into this release
  - Run post-install or post-upgrade hooks and track each of the hooks till successful or failed
- method of applying changes - **2-way merge**
- if deploy failed (2-way-merge)

Werf | 3-way-merge

1. *PATCH = MERGE(PrevConfig + CurrState + NewConfig)* per resoruce
2. Werf Migration plan
    - Annotation mode (`debug.werf.io/repair-patch:<>`)
    - Compatibility mode
    - Full mode

Werf | It's time to start

Когда конфигурация готова:

1. `werf build-and-publish --stages-storage :local --tag-custom reddit --images-repo :minikube`
2. `werf deploy --stages-storage :local --images-repo :minikube --tag-custom reddit --env staging`
3. `werf dismiss --env staging --with-namespace`

Werf | Demo

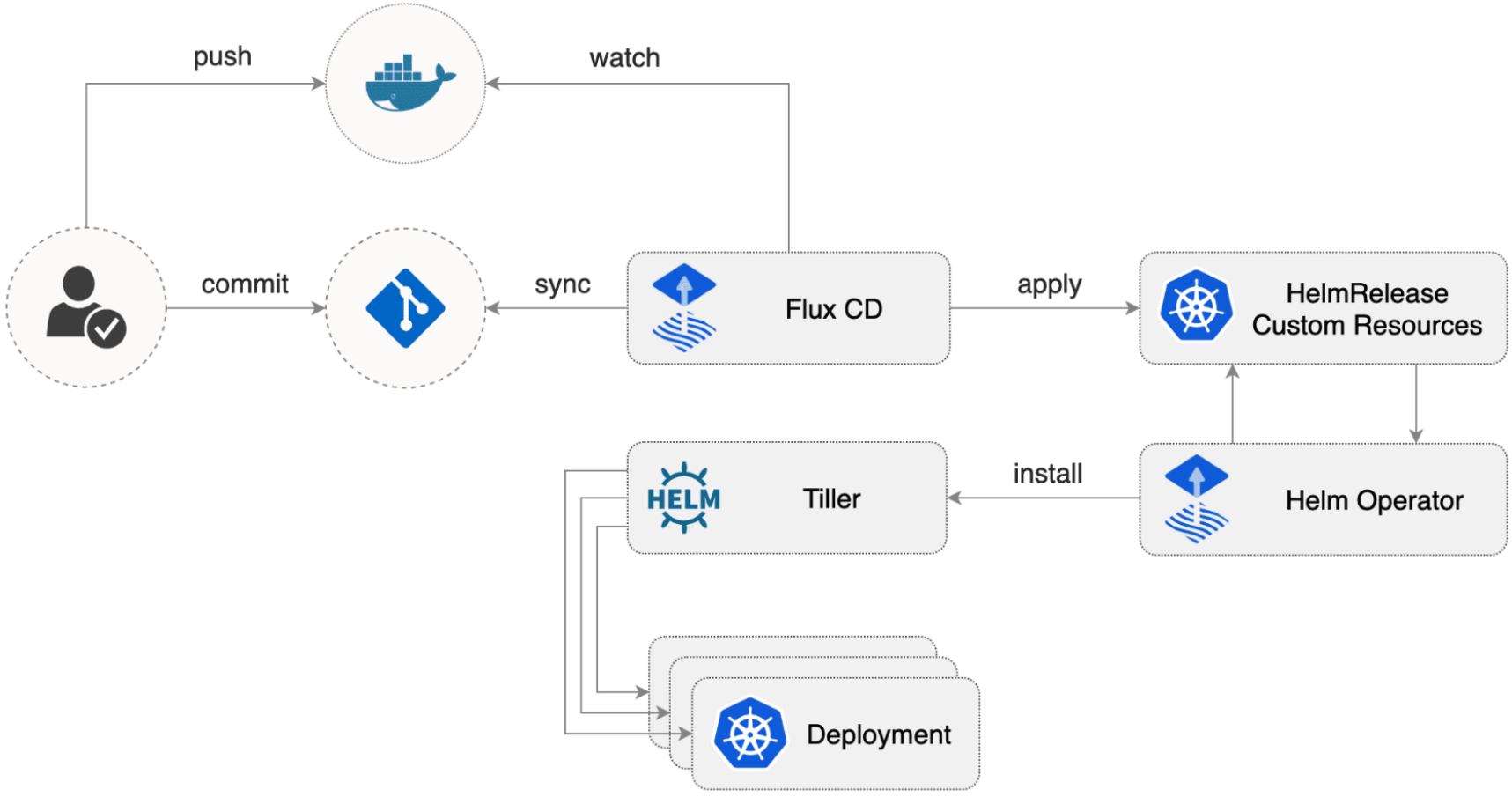
Подводя итоги

- Функциональность CI
- Поддержка локальной разработки: JenkinsX / Werf / Skaffold
- Возможность использования темплейтинга (какого?)
- Observability features
- User friendly
  - UI
- Поддержка современных моделей деплоя из коробки
- **Стабильность**

Не попали в обзор

- kubedex/helm-controller
- ekntoncd/pipelines
- etc...
- аспекты безопасности и работы инструментов с секретами

Flux | Demo


push
watch
commit
sync
Flux CD
apply
HelmRelease
Custom Resources
HELM
Tiller
install
Helm Operator
Deployment

Субъективное мнение

- Argo CD - прост, удобен и незамысловат. Решает конкретную задачу
- Argo Rollouts - только набирает обороты
- JenkinsX - решает почти все, но очень сложен в освоении
- Spinnaker - тяжело, больно, непонятно
- Werf - слишком много пытается решить проблем
- Prow - незаменим в своей области
- Flux - удобно, зрело. Пионер направления
- Skaffold