KubeCon | CloudNativeCon

Europe 2020

Virtual

Kind-ly Validating your K8s Apps Automatically per PR

Sarah Khalife & Grant Griffiths

GitHub

Sarah Khalife

@skhalife

@_skhalife

Field Solutions Engineer

- 11 months @ GitHub
- Previously Cloud Apps & Platform Engineer @ GE
- Focused on inner source, automation, and the SDLC
- For fun, I enjoy volleyball, travel, and the beach! 🏐✈️🏝️

portworx

Grant Griffiths

@ggriffiths

@griffithsgrant

Software Engineer

- 1.5 years @ Portworx
- Previously Data Services & Platform Engineer @ GE
- Contributor to SIG Storage and Kubernetes-CSI
- For fun, I like climbing, soccer, and surfing 🧗⚽🌊

Overview ←

kind

Automation & CI

Testing

Takeaways

Overview

kind ←

Automation & CI

Testing

Takeaways

Overview

kind

Automation & CI ←

Testing

Takeaways

Overview

kind

Automation & CI

Testing ←

Takeaways

KubeCon | CloudNativeCon
Europe 2020
Virtual

Overview

kind

Automation & CI

Testing

Takeaways ←

Overview

Overview ←
kind
Automation & CI
Testing
Takeaways

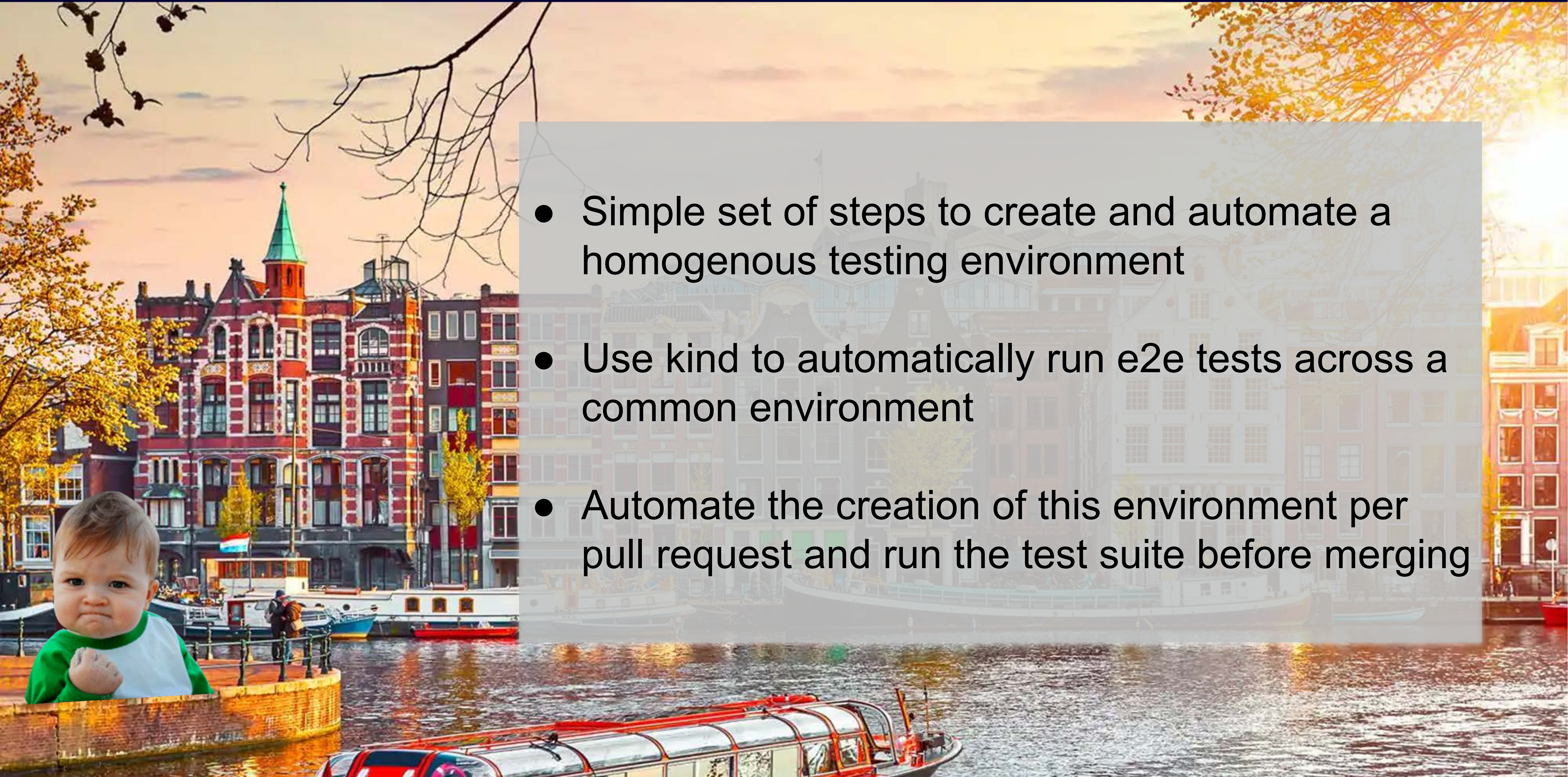
Motivation

We want to be able to collaborate successfully in both our internal & open source projects.

What are the challenges?

- Collaboration needs to be **transparent**, **consistent**, and **rigorous**
- **Testing k8s application varies** per developer environment
- **Time and resource consuming** to constantly spin up k8s clusters

What we’ll cover



- Simple set of steps to create and automate a homogenous testing environment
- Use kind to automatically run e2e tests across a common environment
- Automate the creation of this environment per pull request and run the test suite before merging

kind

Overview

kind ←

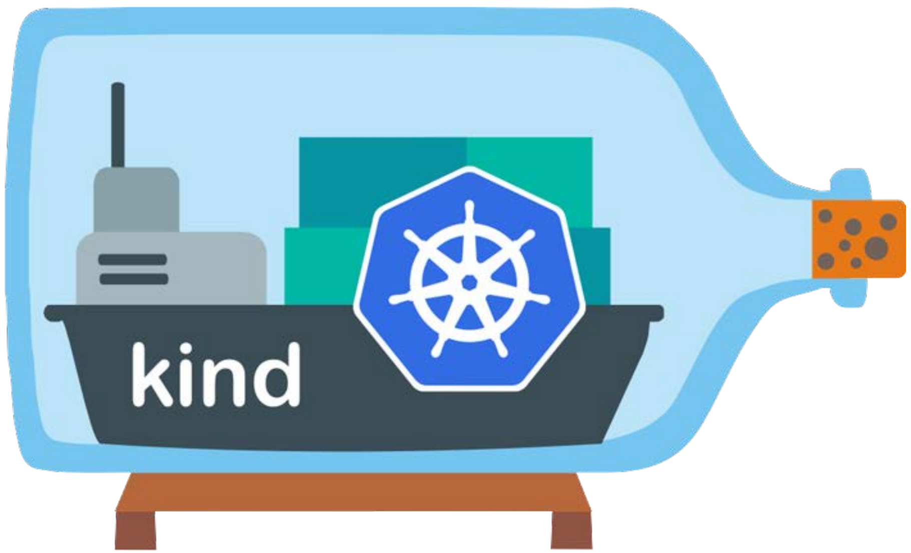
Automation & CI

Testing

Takeaways

What is kind?

- Kubernetes in Docker
- Similar in usage to Minikube and k3s
- Great for local testing
- kind.sigs.k8s.io



How to use kind

Could not be any easier!

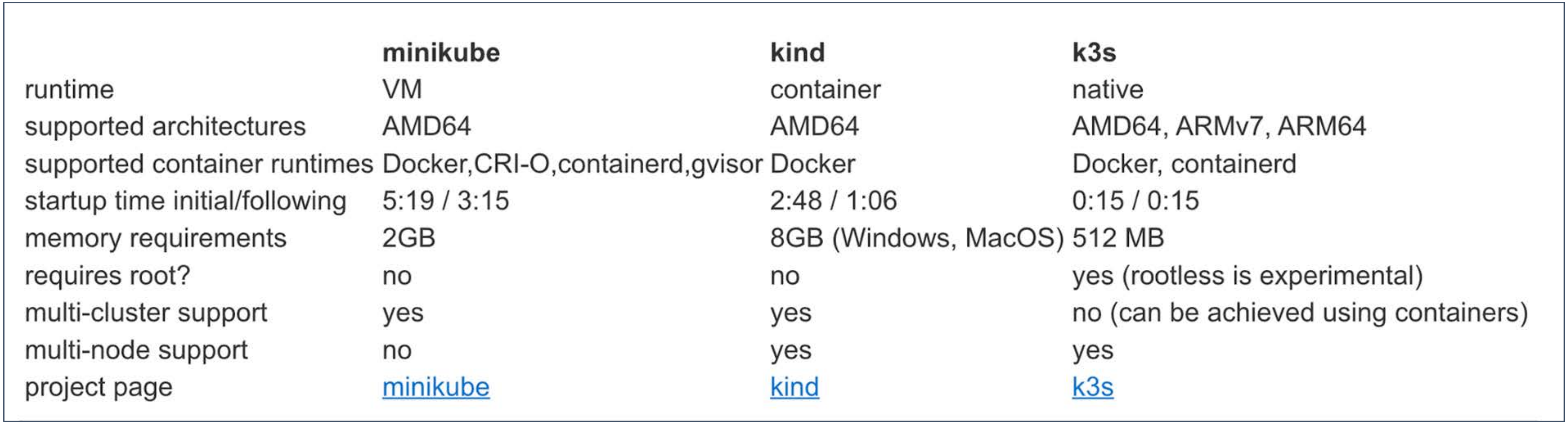
```
➜  ~ brew install kind
➜  ~ kind create cluster
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.18.2) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Not sure what to do next? 😅  Check out https://kind.sigs.k8s.io/docs/user/quick-start/
```

kind vs. others

kind - quick, easy to deploy, low barrier to entry, great for testing

k3s - even quicker, minimal version of k8s

minikube - single node k8s, low barrier to entry



	minikube	**kind**	**k3s**
runtime	VM	container	native
supported architectures	AMD64	AMD64	AMD64, ARMv7, ARM64
supported container runtimes	Docker,CRI-O,containerd,gvisor	Docker	Docker, containerd
startup time initial/following	5:19 / 3:15	2:48 / 1:06	0:15 / 0:15
memory requirements	2GB	8GB (Windows, MacOS)	512 MB
requires root?	no	no	yes (rootless is experimental)
multi-cluster support	yes	yes	no (can be achieved using containers)
multi-node support	no	yes	yes
project page	minikube	kind	k3s

https://brennerm.github.io/posts/minikube-vs-kind-vs-k3s.html

KubeCon | CloudNativeCon Europe 2020 Virtual

Automation & CI

Overview
kind
Automation & CI ←
Testing
Takeaways

Goals

Run our validation test on new code changes

- Reduce amount of bugs
- Validate before it hits prod

Block merge to prod if tests have failed

- Don’t introduce broken code into main branch
- Hold dev accountable to fix issue

Setup a pre-configured environments when testing

- Catch problems with your application logic systematically
- Maintain consistency in results

Automate everything with event triggers

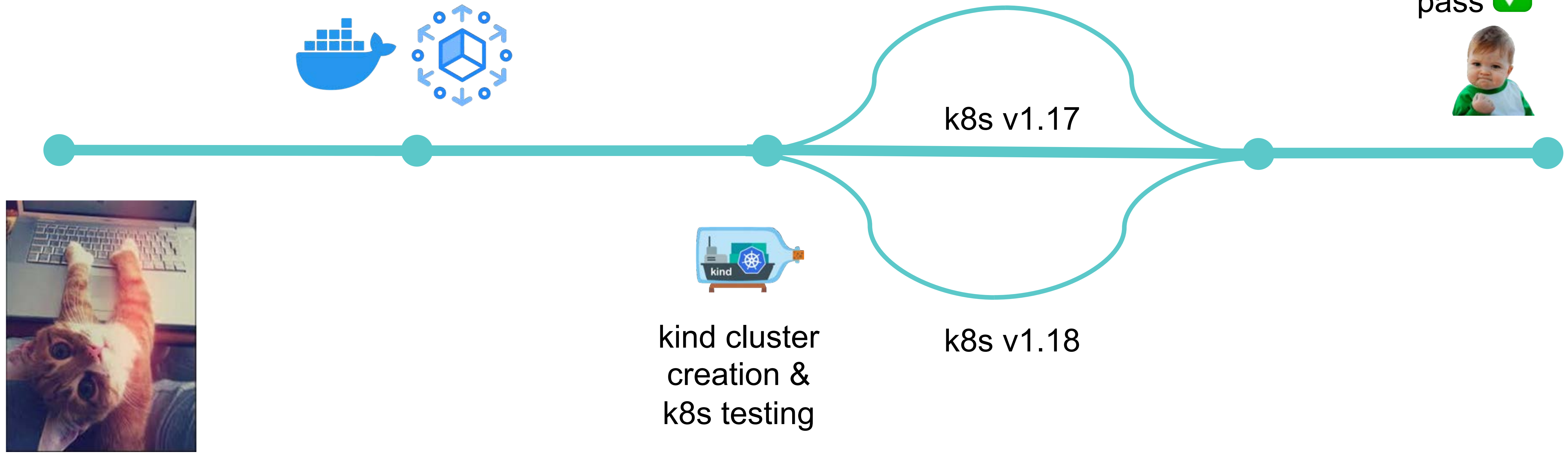
- Increase frequency of test runs
- Repeatable and generally more efficient

Start kind with GitHub Actions

```
name: "Create cluster using kind"
on: [pull_request]

jobs:
  kind:
    runs-on: ubuntu-latest
    steps:
    - uses: engineerd/setup-kind@v0.4.0
```

github.com/marketplace/actions/kind-kubernetes-in-docker-action

Automation workflow for demo



Dev introduces code change, and creates a PR

Build & Publish

kind cluster creation & k8s testing

k8s v1.16

k8s v1.17

k8s v1.18

Merge PR when test pass ✅

Testing

Overview

kind

Automation & CI

Testing ←

Takeaways

When to use integration & e2e tests

- Dependency between apps
- Code interacts with k8s objects
- May not always be needed
  - resource & time intensive
  - unit tests can cover business logic

Sample problem - CSI Driver Testing

- Used by Kubernetes-CSI team github.com/kubernetes-csi/csi-release-tools
- Basic flow
  - Create kind cluster
  - Deploy a sample CSI driver & sidecars
  - Run tests

Sample problem - Portworx e2e testing

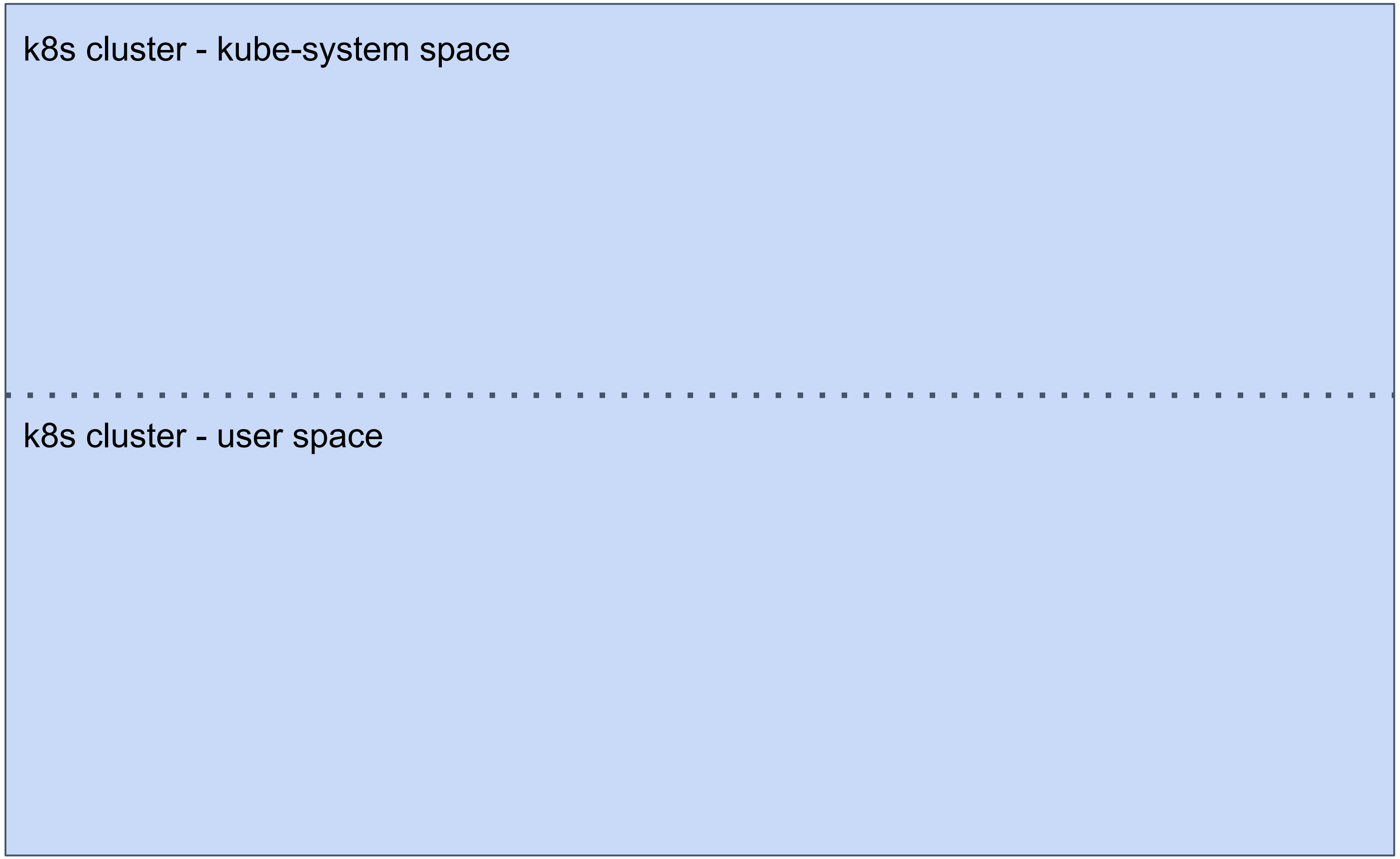
- **Portworx** - Enterprise Storage Platform for k8s
- **Openstorage** - the open source control plane for Portworx
- Test a feature called **Portworx Security**
- Prevents unauthenticated users from accessing the platform
- We will test using the Portworx CSI Driver

portworx

Example - Portworx e2e Test

Automation steps

1. Create kind cluster



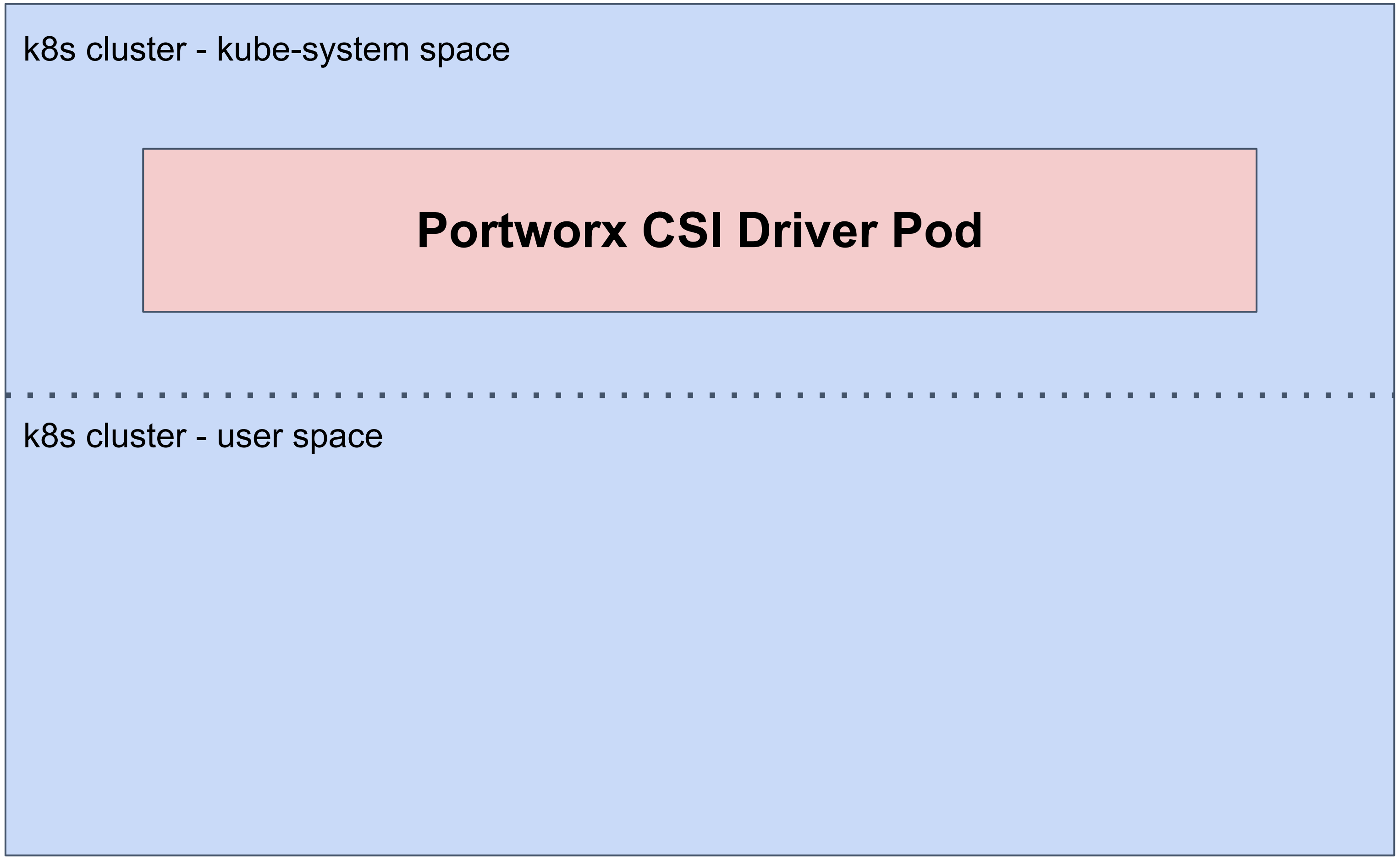
k8s cluster - kube-system space

k8s cluster - user space

Example - Portworx e2e Test

Automation steps

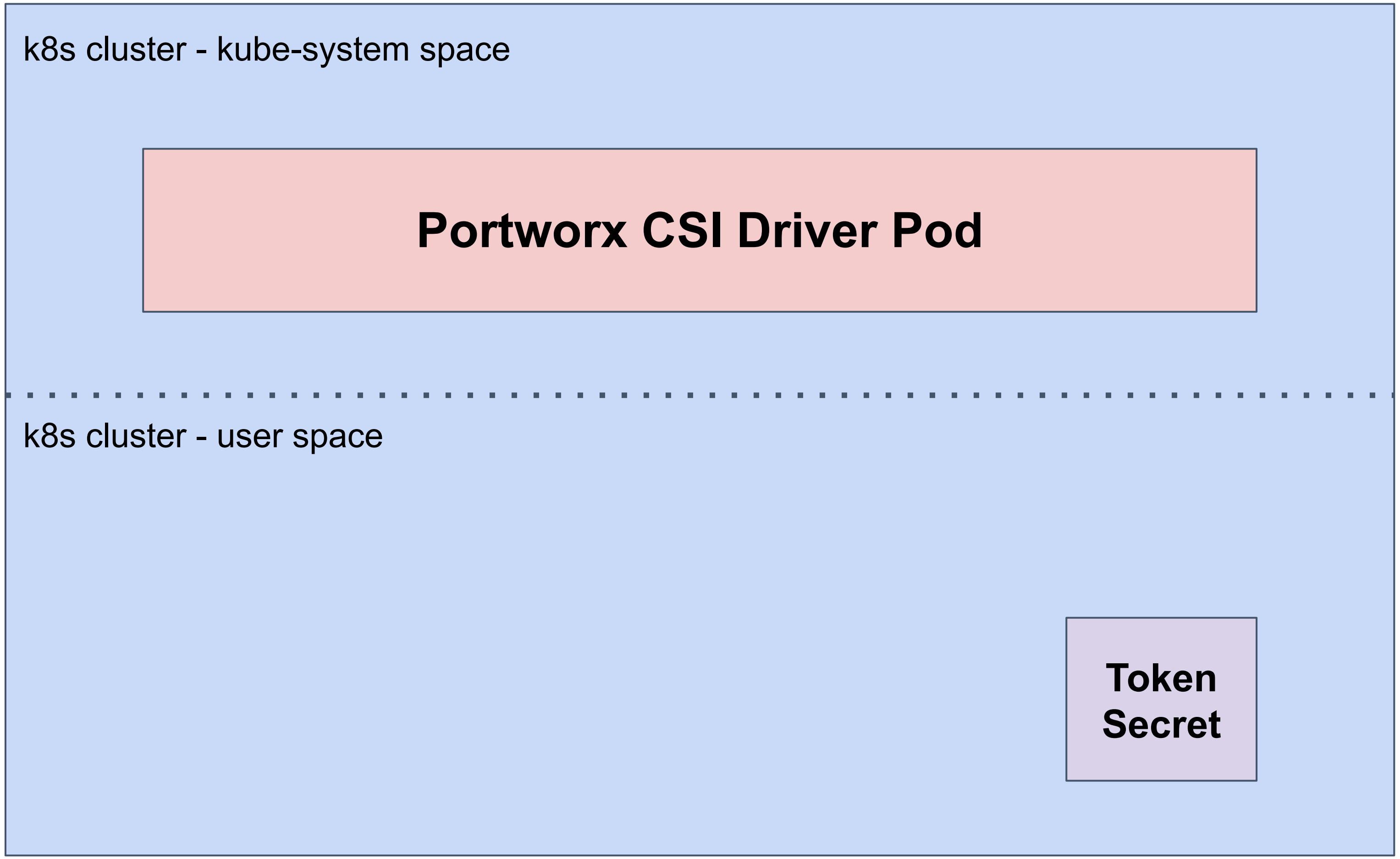
1. Create kind cluster
2. Deploy Portworx CSI Driver



Example - Portworx e2e Test

Automation steps

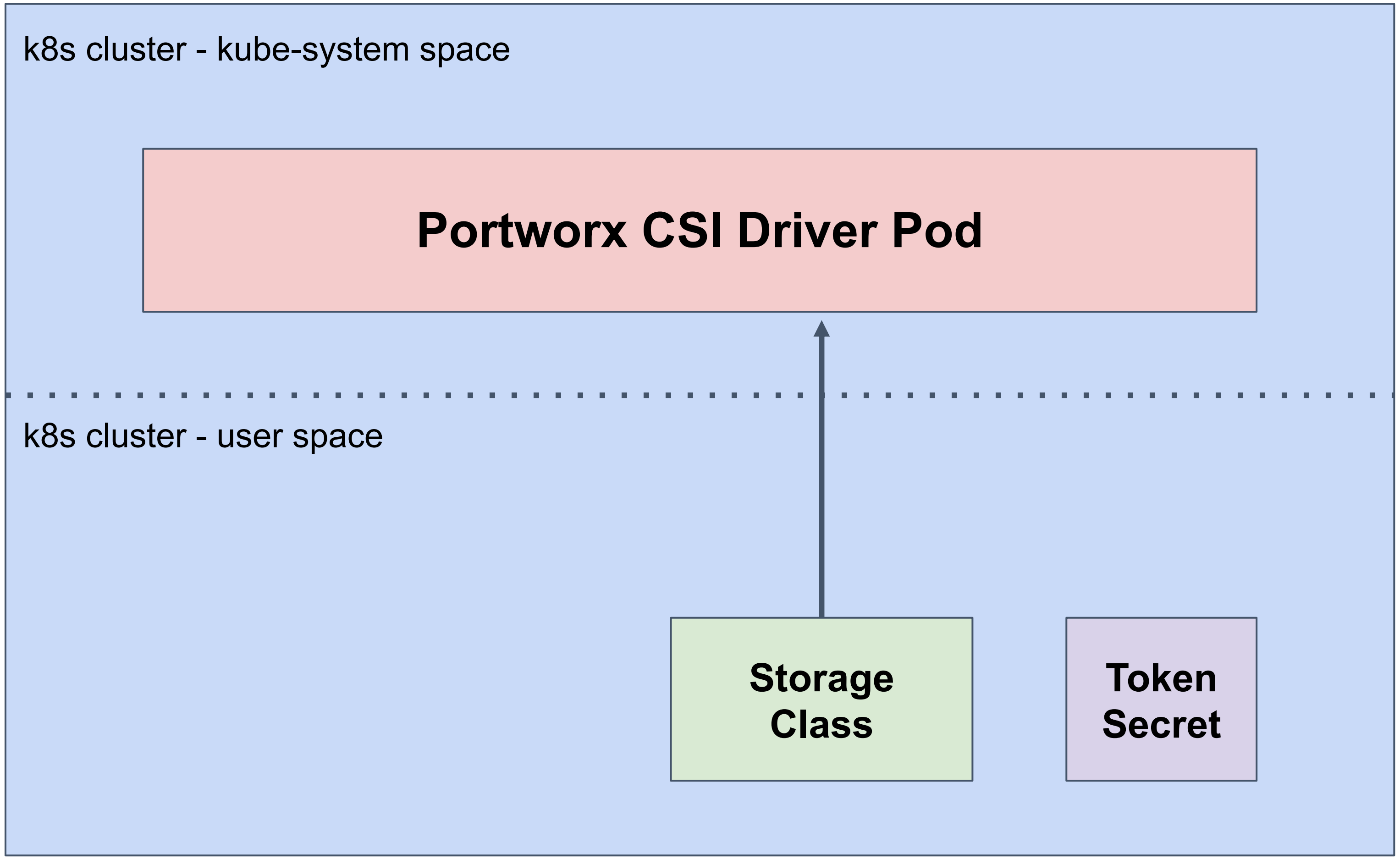
1. Create kind cluster
2. Deploy Portworx CSI Driver
3. Create token secret



Example - Portworx e2e Test

Automation steps

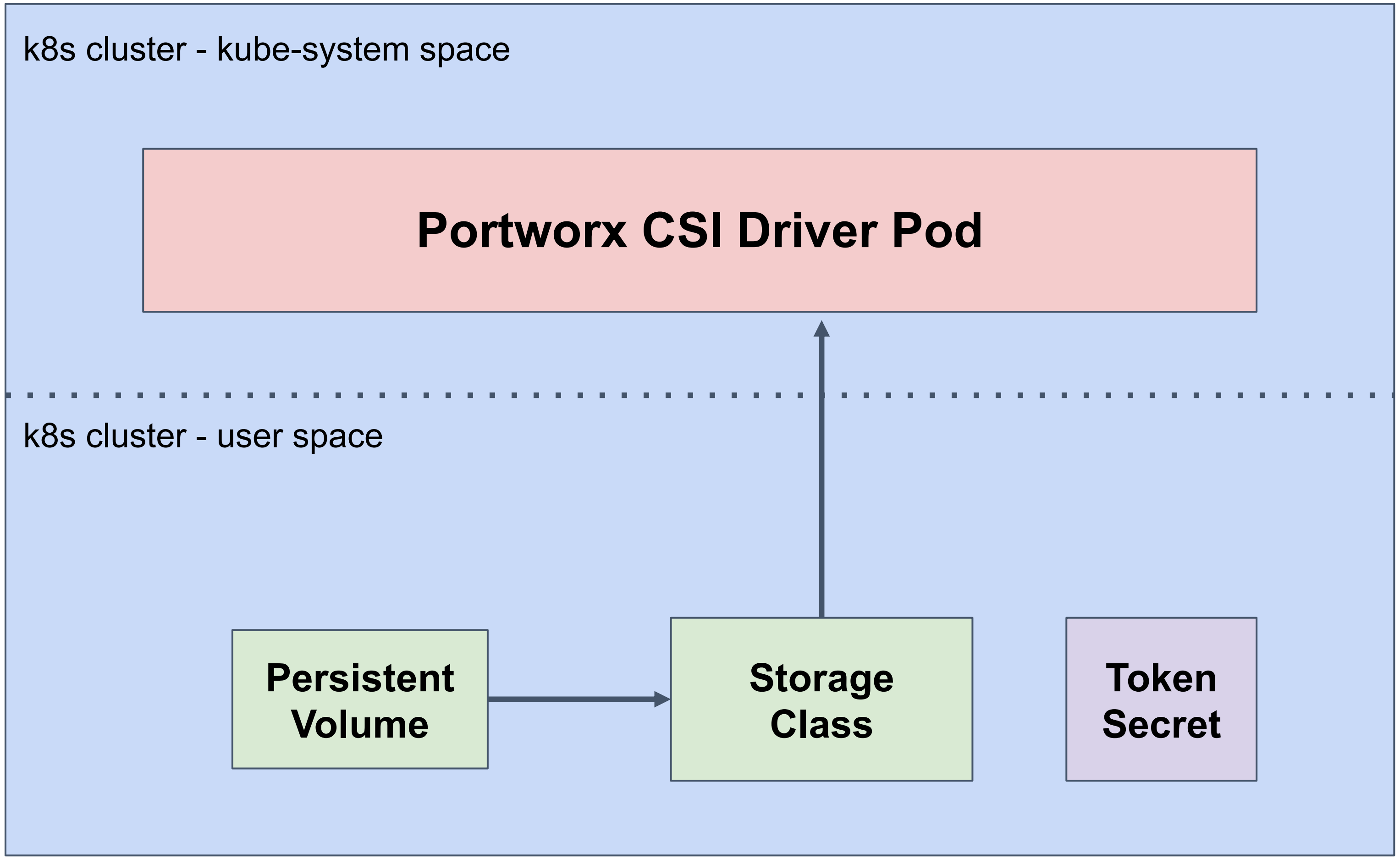
1. Create kind cluster
2. Deploy Portworx CSI Driver
3. Create token secret
4. Create storage class



Example - Portworx e2e Test

Automation steps

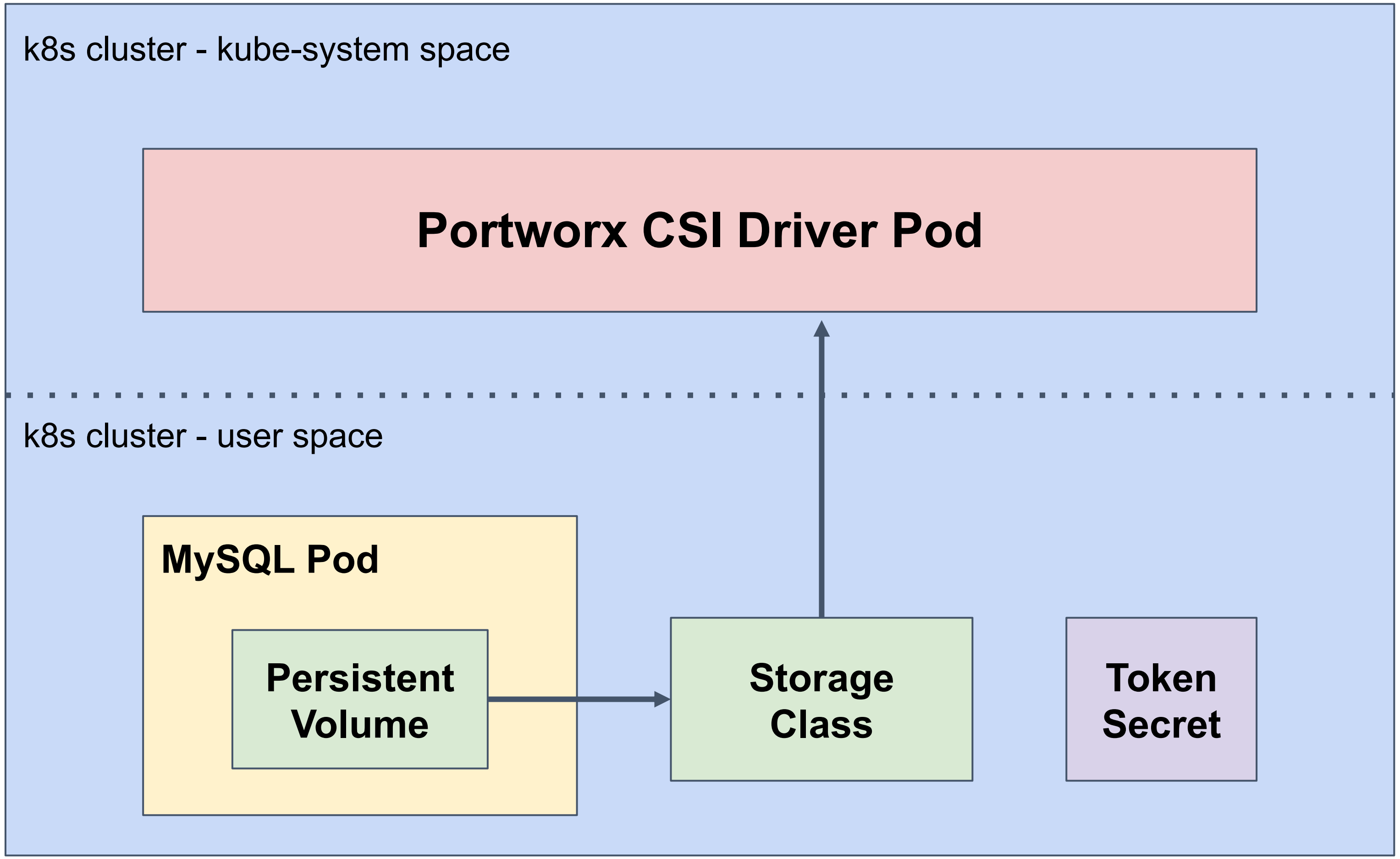
1. Create kind cluster
2. Deploy Portworx CSI Driver
3. Create token secret
4. Create storage class
5. Create persistent volume



Example - Portworx e2e Test

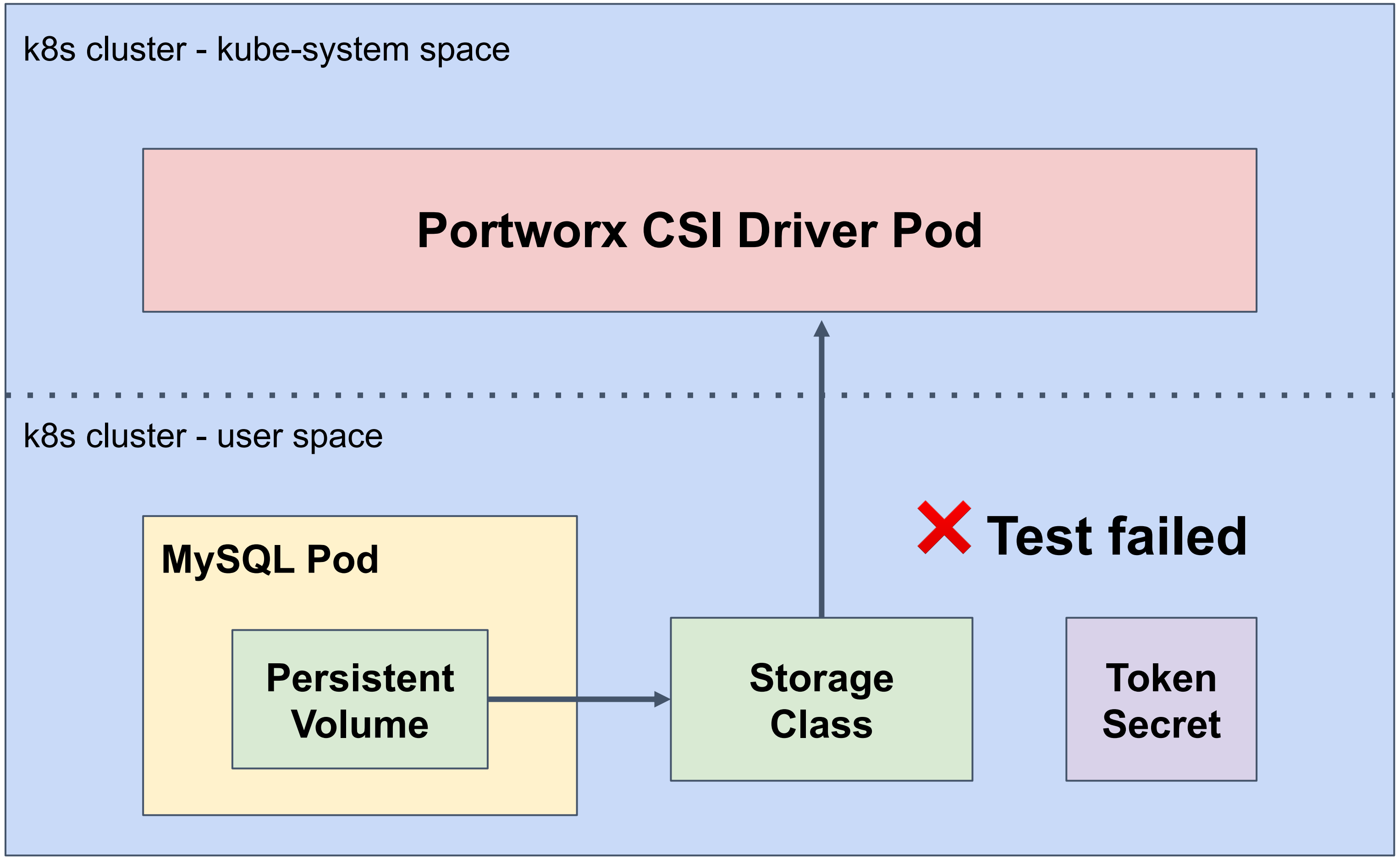
Automation steps

1. Create kind cluster
2. Deploy Portworx CSI Driver
3. Create token secret
4. Create storage class
5. Create persistent volume
6. Create pod with persistent volume



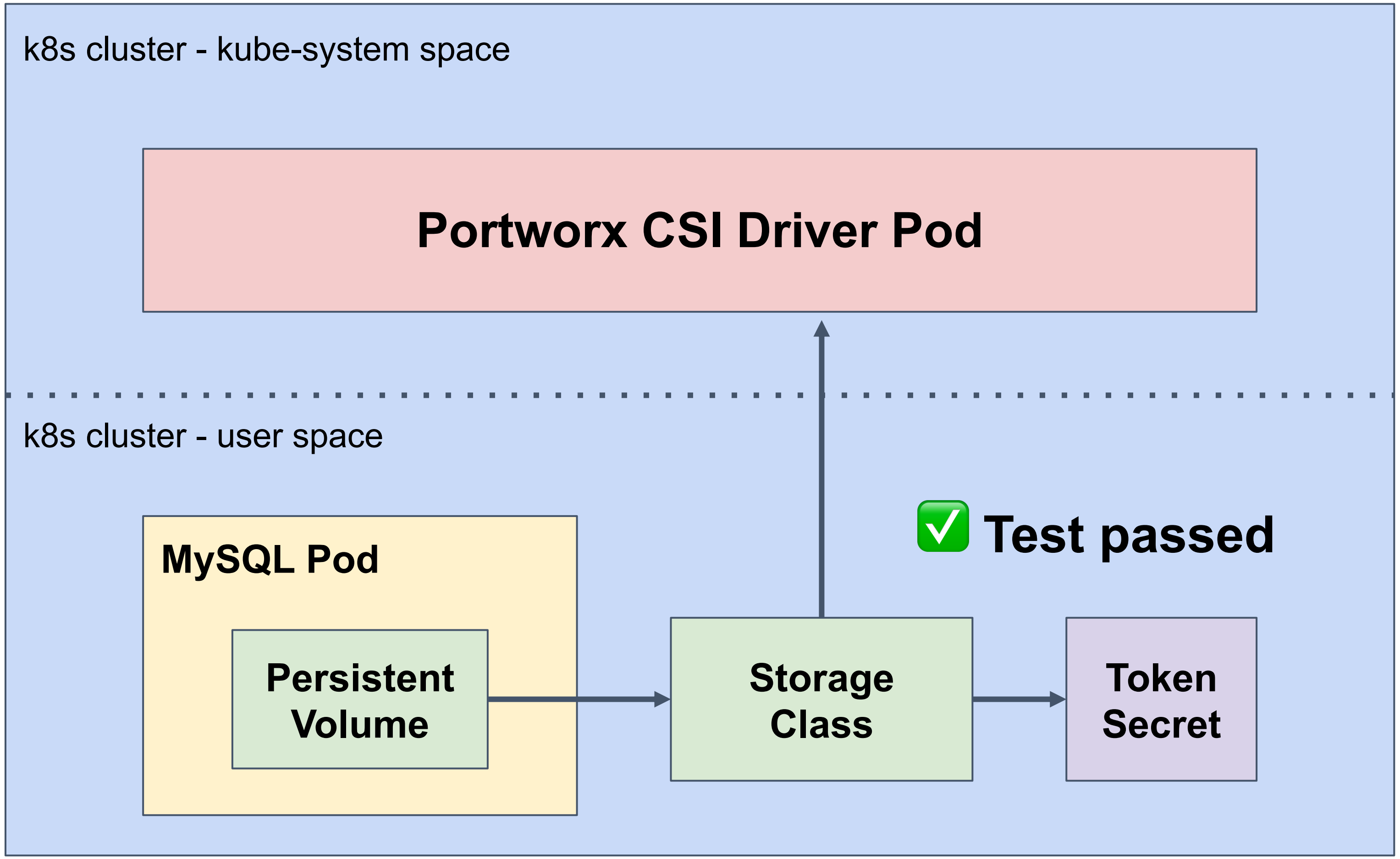
Example - Portworx e2e Test

Test: Token secret reference must be valid in order to create and use a Portworx volume



Example - Portworx e2e Test

Test: Token secret reference must be valid in order to create and use a Portworx volume



🚀

Live Demo

Takeaways

No more manual work

Automate based on event triggers - decide what type of events best suit the job you want to run:

- *Should I run this test on every push to main?*

- *When should I deploy my app to GKE?*

```
name: Docker Image CI

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
```

```
name: Build and Deploy to GKE

on:
  release:
    types: [created]
```

No more manual work

Set up your workflow - decide what jobs you need to run:

- *I need to build my docker image & push it to DTR*

```yaml
jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Build the Docker image
      run: make docker-build-osd
```

- *I need a kind job to spin up k8s & run my test*

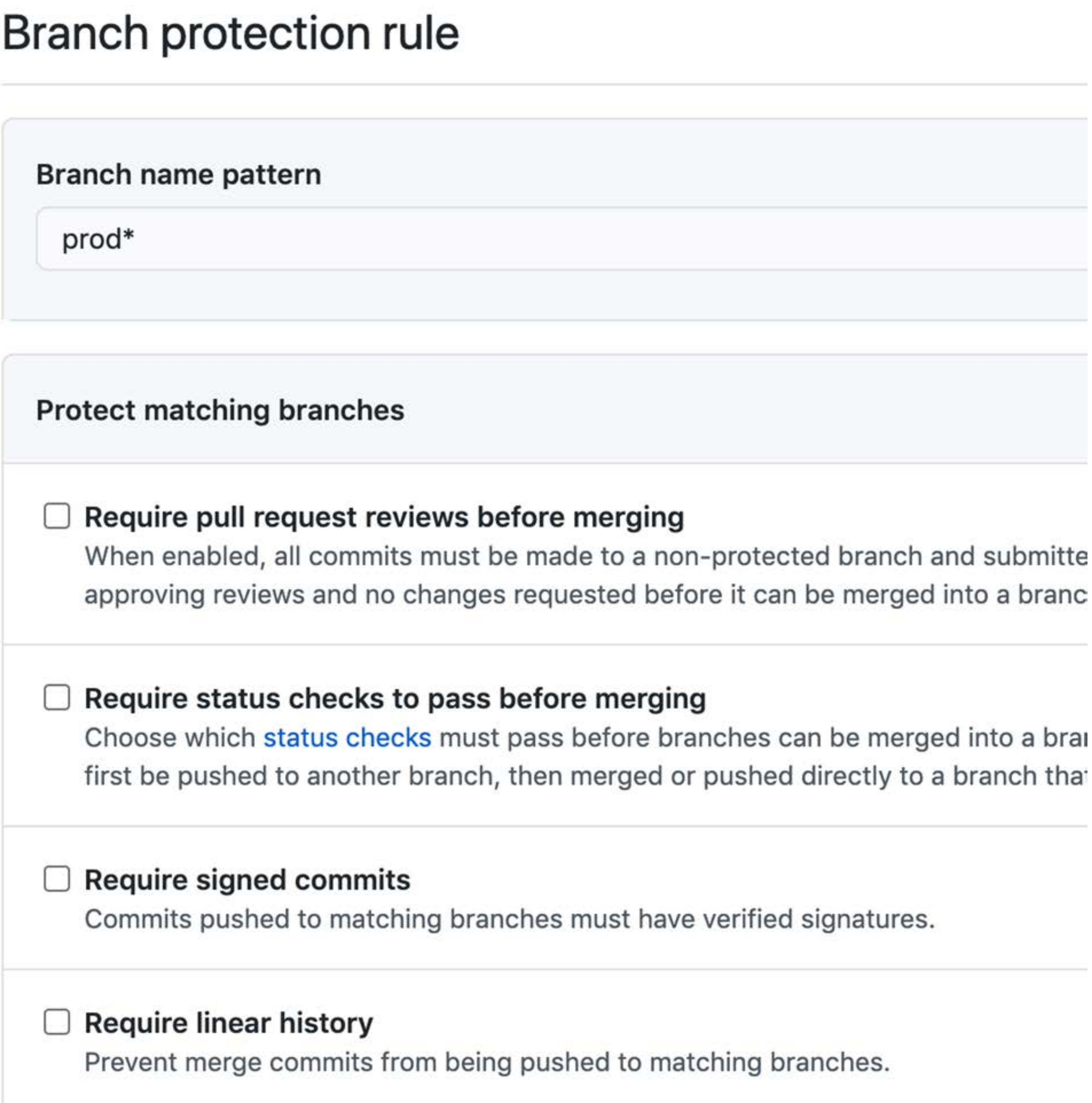
- *kind needs docker build to succeed.*

```yaml
jobs:
  kind:
    needs: build
    runs-on: ubuntu-latest
    strategy:
      matrix:
        k8s-version: ['v1.16', 'v1.17', 'v1.18']
```

No more manual work

Setup Protected branches - choose the rules that should apply:

- Require code reviews
- Have status checks based on workflows
- Disable force push



Branch protection rule

Branch name pattern

prod*

Protect matching branches

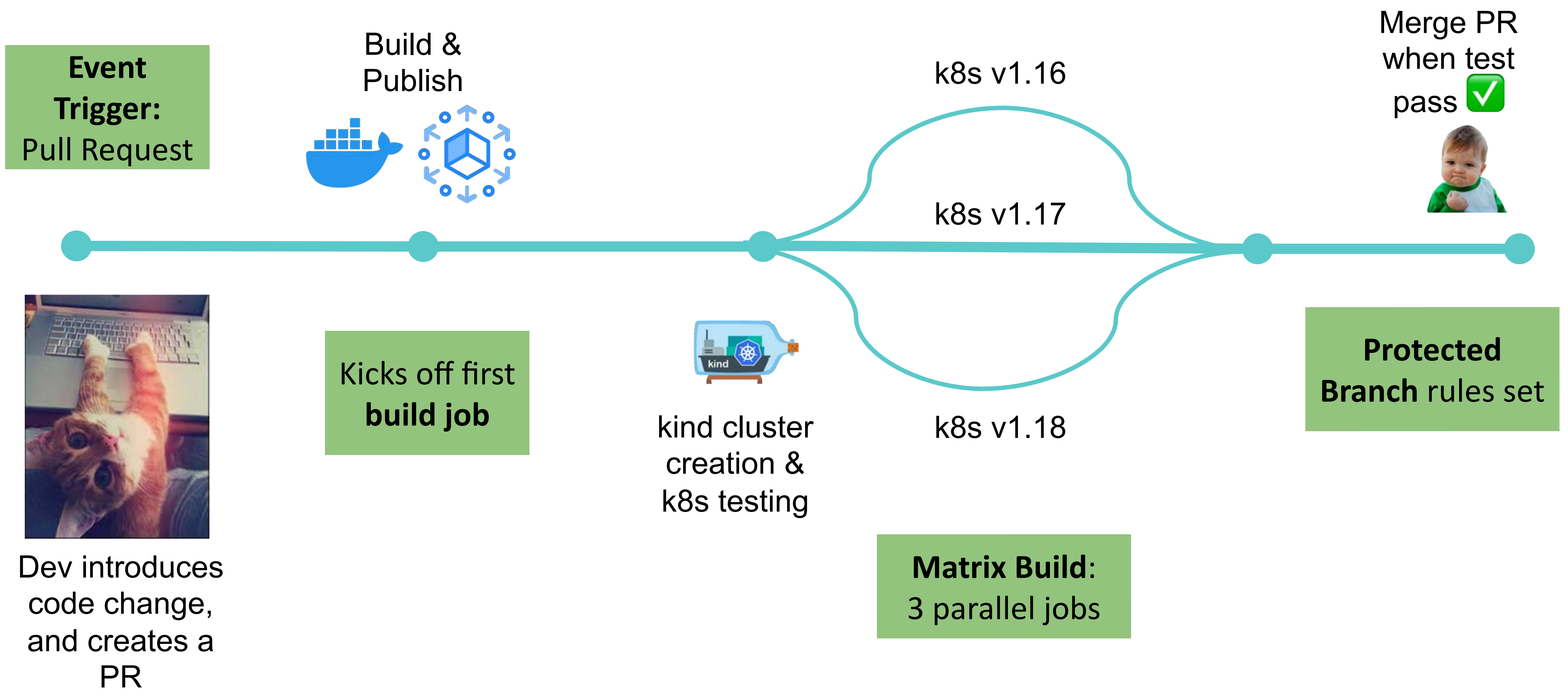
□ **Require pull request reviews before merging**
When enabled, all commits must be made to a non-protected branch and submitte approving reviews and no changes requested before it can be merged into a branc

□ **Require status checks to pass before merging**
Choose which status checks must pass before branches can be merged into a brai first be pushed to another branch, then merged or pushed directly to a branch tha

□ **Require signed commits**
Commits pushed to matching branches must have verified signatures.

□ **Require linear history**
Prevent merge commits from being pushed to matching branches.

Recap



Event Trigger:
Pull Request

Dev introduces code change, and creates a PR

Build & Publish

Kicks off first **build job**

kind cluster creation & k8s testing

k8s v1.16

k8s v1.17

k8s v1.18

Matrix Build:
3 parallel jobs

Merge PR when test pass ✅

Protected Branch rules set

Lessons Learned

WHAT

- Breaking down the "*build*" and "*kind*" jobs in two separate ones
- Triggering e2e test workflows only on PRs to the master branch
- Use matrix builds to run your tests to validate against multiple k8s versions
- Test against the same image in all workflows

WHY

- Simplified builds, cleaner workflows, & more efficient runs
- Will otherwise be inefficient, using up unnecessary resources
- Concurrent testing with same automation script
- Saves time and resources, can always audit and refer back to specific image

Q&A

KubeCon | CloudNativeCon

Europe 2020

Virtual

KEEP CLOUD NATIVE CONNECTED