

# What Developers want (and what platform people can do about it)

40 min



# Your hosts today



Developer Advocate at Codefresh

**Kostis Kapelonis**

✉ [kostis@codefresh.io](mailto:kostis@codefresh.io)



Lead Architect at Port

**Mor Paz**

✉ [mor@getport.io](mailto:mor@getport.io)

# Agenda

- Intro - information overload
- Demo 1
- Give developers what they need
- Demo 2
- Deep dive into the mechanics
- Demo 3

# People first

# People



**Developer**

Focus is on writing code and shipping features



**Operations/Ops**

Focus is on running the systems and giving infrastructure to developers



**DevEx**

Focus is on day-2-day productivity, onboarding and removing obstacles

# Developers

- Want to ship features
- Move fast
- Don't really want to learn K8s
- Want to follow commit to prod
- Want easy way to create env
- Like clicking buttons



# SRE/Operations/DevOps/platform/infra

- Want healthy systems
- Move slow
- Hate open tickets
- Prefer automation
- Want to help developers
- Ideally everything is self-service



# DevEX/Team lead/CTO etc.

- Care about development onboarding
- Monitor sprint velocity
- Responsible for day-to-day operations and blockers
- Hate “walls”, silos and lengthy drop-off procedures





# Information overload

# Simple booking example (5 options)

 Book a flight

Round trip

---

 Departing from \*

---

 Arriving at \*

---

Travel dates

 12 July – 29 July



Passengers

1 adult

# Complex booking example (20 options)

✈ Book a flight

Round trip

✍ Departing from \*

✍ Arriving at \*

Travel dates

📅 12 July – 29 July



Passengers

1 adult

✍ Choose pilot

✍ Choose Airplane

✍ Terminal (depart)

✍ Terminal (arrive)

✍ Meal catering company

✍ Fuel company

✍ Fuel amount

✍ Choose Co-pilot

✍ Choose ground crew

✍ Choose air-hostess crew

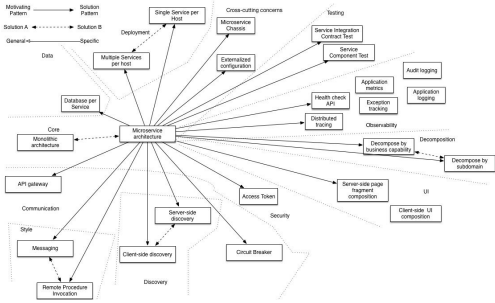
✍ Air-control hand-off

✍ Cleaner crew

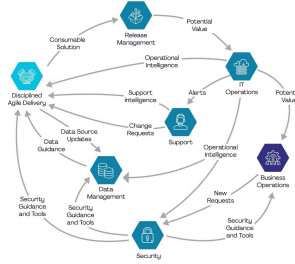
✍ Baggage handlers

✍ Insurance

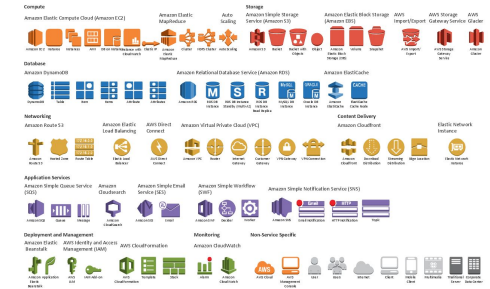
# Microservices



# Environment types



# Managed services



# Infra management



Too much cognitive load for developers (and DevOps alike)

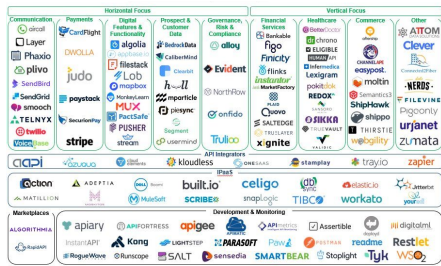
# Clouds



# Code shipment



# DevTools



# Databases & ETL



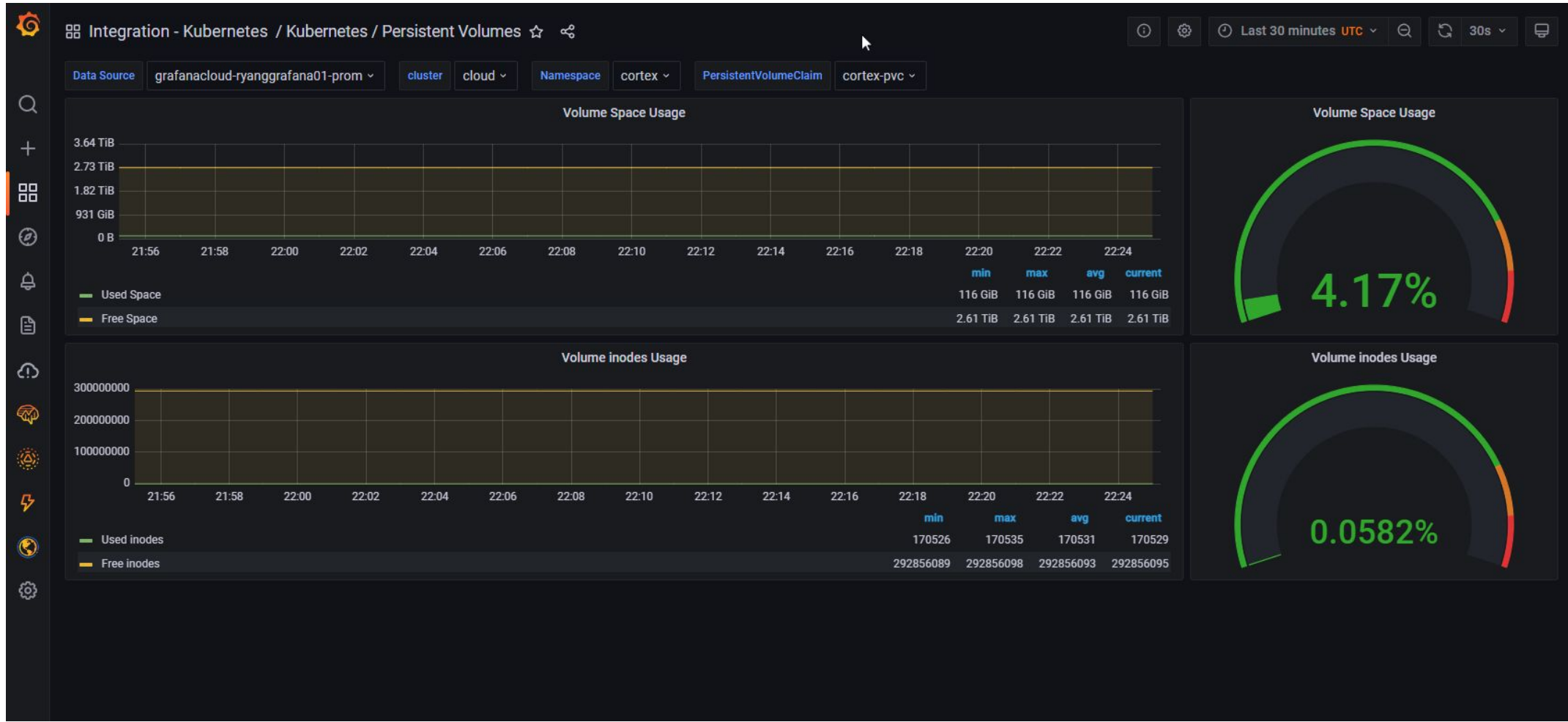
# Permissions



# Configurations



# Is this what developers want?



# Is this what developers want?

The screenshot displays the Port application management interface for an application named 'guestbook'. The interface is organized into several sections:

- Header:** Shows 'Applications / guestbook' and 'APPLICATION DETAILS'.
- Navigation Bar:** Contains buttons for 'APP DETAILS', 'APP DIFF', 'SYNC', 'SYNC STATUS', 'HISTORY AND ROLLBACK', 'DELETE', and 'REFRESH'.
- APP HEALTH:** Displays a green heart icon and the word 'Healthy'.
- CURRENT SYNC STATUS:** Shows a green checkmark, 'Synced', and 'To HEAD (53e28ff)'. A 'MORE' link is available.
- LAST SYNC RESULT:** Shows a green checkmark, 'Sync OK', and 'To 53e28ff'. It includes details: 'Succeeded 3 minutes ago (Fri Nov 05 2021 15:19:49 GMT-0500)', 'Author: May Zhang <may\_zhang@intuit.com>', and 'Comment: feat: update helm samples to use helm3 (#78)'. A 'MORE' link is also present.
- Resource Diagram:** A central diagram shows the application's architecture. It starts with the 'guestbook' application (13 minutes) which connects to a 'svc' (3 minutes) and a 'deploy' (3 minutes, rev:1). The 'svc' connects to a 'guestbook-ui' (3 minutes), which in turn connects to an 'ep' (3 minutes). The 'ep' connects to an 'ES' (3 minutes) labeled 'guestbook-ui-khkww' (endpointslice). The 'deploy' connects to an 'rs' (3 minutes, rev:1) labeled 'guestbook-ui-85985d774c'. Finally, the 'rs' connects to a 'pod' (3 minutes, running, 1/1) labeled 'guestbook-ui-85985d774c-dpfb2'.

# Focusing on what matters with Port

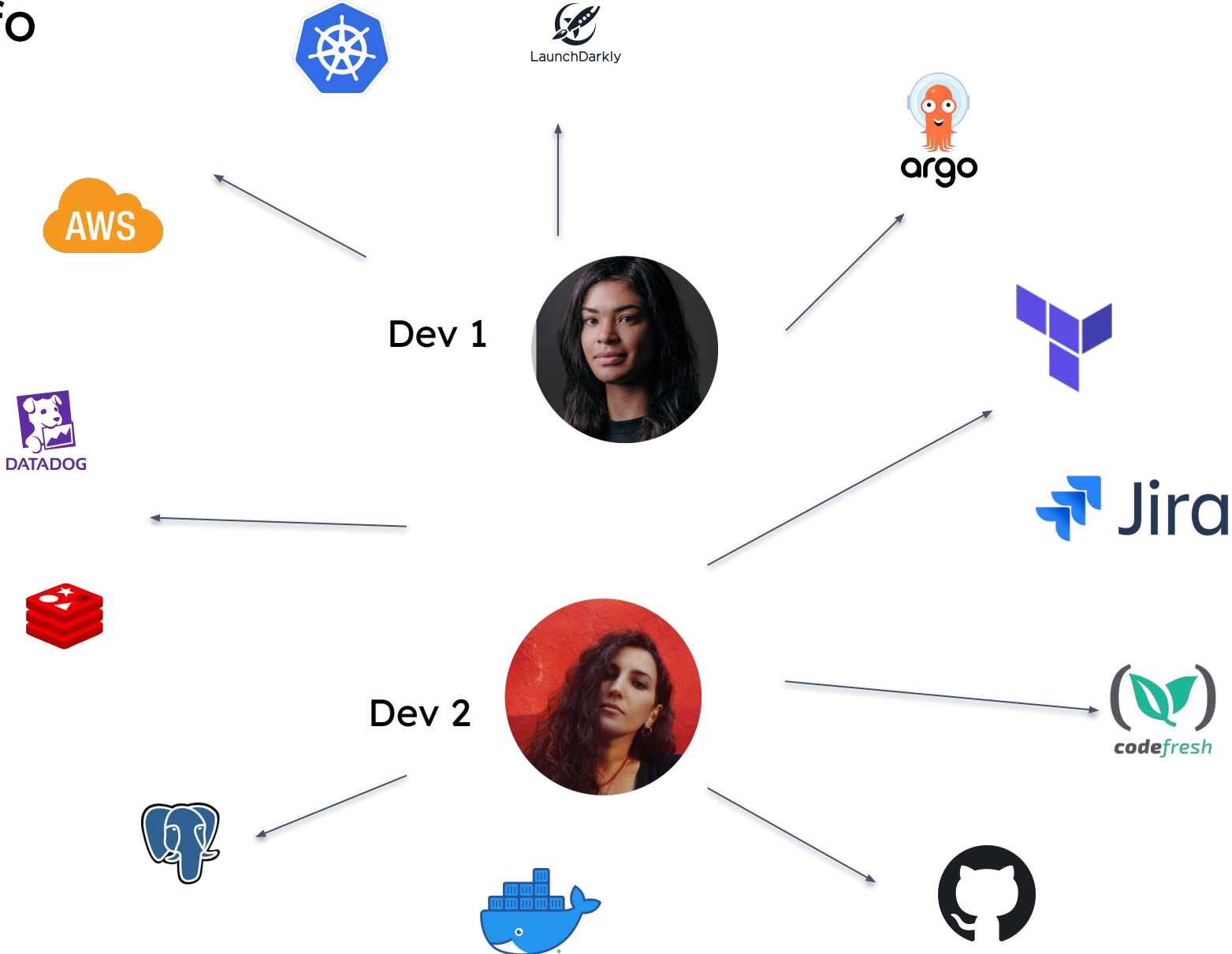
# An Open Internal Developer Portal

The screenshot displays the Port internal developer portal interface. The top navigation bar includes the 'port' logo, 'Home', 'Catalog', 'Self-service', and 'Builder' menus. A left sidebar lists various service categories such as Services, Running Services, Deployments (highlighted), Feature Flags, K8s Clusters, Cloud Permissions, Cloud Resources, Domains, Developer Envs, Packages, Third Parties, Alerts, Vulnerabilities, and Misconfigurations. The main content area is titled 'Deployments' and features a search bar, filter, hide, sort, and group by controls. Below these controls is a table listing deployment records with columns for Domain, Title, Last Update, Deployment Status, User, Workflow URL, Branch URL, and Commit SHA.

Domain	Title	Last Update	Deployment Status	User	Workflow URL	Branch URL	Commit SHA
Shipping	checkout-production	an hour a...	Failed	Michael M...			d47b9e ...
Subscript...	rating-production	an hour a...	In Progress	Ryoko De-...			e6ebf1 ...
Shipping	load-generator-security	an hour a...	Success	Michael M...			6f484 ...
Analytics	order-staging	an hour a...	Success	Francisca ...			af34e ...
Inventory	shipping-production	an hour a...	Success	Michael M...			31b4a ...
Subscript...	payment-staging	an hour a...	Success	Francisca ...			72dd0 ...
Shipping	ads-sandbox	an hour a...	Failed	Francisca ...			b3c80 ...
Shipping	checkout-staging	an hour a...	Failed	Michael M...			fdde1 ...
Inventory	authorization-security	an hour a...	Success	Ryoko De-...			c1f7d6 ...



# Too much info



# Developer portal



Internal developer portal



# Use Cases

-  Cloud Resource Catalog
-  K8s & ArgoCD Visualization
-  Cloud Resource Permissions
-  Ephemeral Environments
-  Microservice Catalog
-  IaC for Developers
-  Package Management
-  On-call and Incident Management

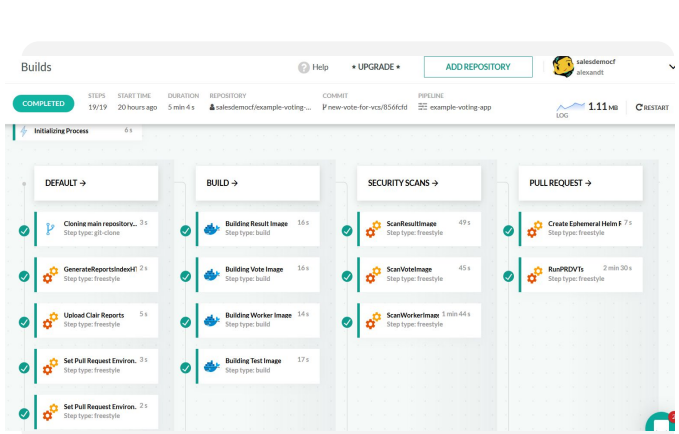
# **Demo 1**

**Creating a model that  
matches your team's needs**

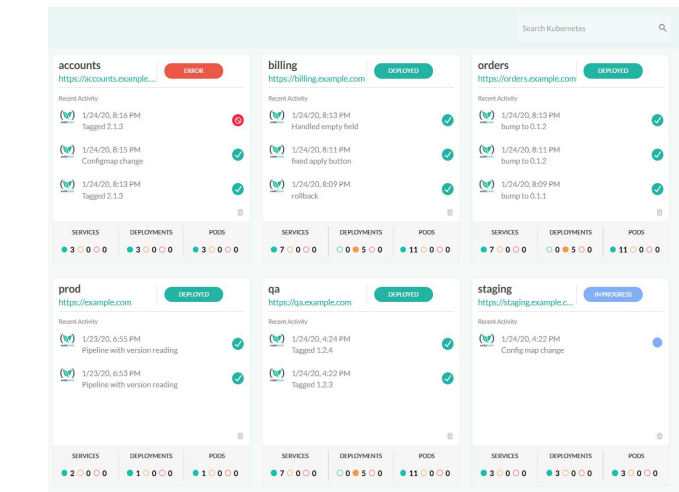
# Behind the scenes

Expose to developers only what they need

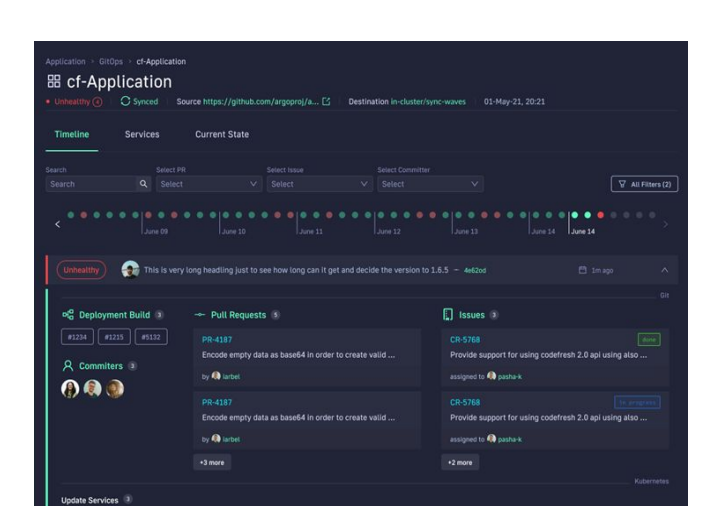
# What Codefresh offers



**Pipelines**  
Implement Continuous Integration



**Deployments**  
Implement Continuous Delivery



**GitOps**  
Enterprise Argo CD

# Codefresh pipelines

- Docker based pipelines
- Native Kubernetes integration
- Native Registry integration
- Argo CD/Rollouts/Workflows/events integration

The screenshot displays a Codefresh pipeline execution interface for a pull request preview. The pipeline is titled "pull-request-preview" and is in a "COMPLETED" state, having finished 6 minutes and 24 seconds ago. The build size is 166.87kB. The pipeline is organized into three stages: PREPARE, VERIFY, and DEPLOY. Each stage contains several steps, all of which are marked as successful with green checkmarks.

Stage	Step Name	Step Type	Duration
PREPARE →	Cloning repository	git-clone	2 s
	Compile/Unit test	freestyle	30 s
	Building Docker Image	build	13 s
VERIFY	Source security scan	freestyle	44 s
	Container security scan	freestyle	12 s
	Integration tests	freestyle	1 min 34 s
	Sonar Scan	freestyle	1 min 17 s
DEPLOY	Cloning repository	git-clone	3 s
	Deploying Helm Chart	helm	17 s
	Adding comment on PR	kostis-codefresh/github-pr-comment	16 s
	Smoke tests	freestyle	42 s

# Building/Pushing an image

## GitHub Actions

```
- name: Log in to Docker Hub
  uses: docker/login-action@f4ef78c080cd8ba55a85445d5b36e214a81df20a
  with:
    username: ${ secrets.DOCKER_USERNAME }}
    password: ${ secrets.DOCKER_PASSWORD }}

- name: Extract metadata (tags, labels) for Docker
  id: meta
  uses: docker/metadata-action@9ec57ed1fcd8bf14dcef7dfbe97b2010124a938b7
  with:
    images: my-docker-hub-namespace/my-docker-hub-repository

- name: Build and push Docker image
  uses: docker/build-push-action@3b5e8027fcad23fda98b2e3ac259d8d67585f671
  with:
    context: .
    file: ./Dockerfile
    push: true
    tags: ${ steps.meta.outputs.tags }}
    labels: ${ steps.meta.outputs.labels }}
```

## Codefresh

```
BuildMyImage:
  title: Building My Docker image
  type: build
  image_name: my-app-image
  registry: dockerhub
  working_directory: '${ main_clone }'
  tag: 1.0.1
```



# Deploying to Kubernetes

## GitHub Actions

```
- name: Configure AWS credentials
  uses: aws-actions/configure-aws-credentials@v1
  with:
    aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
    aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
    aws-region: us-east-2

- name: Login to Amazon ECR
  id: login-ecr
  uses: aws-actions/amazon-ecr-login@v1

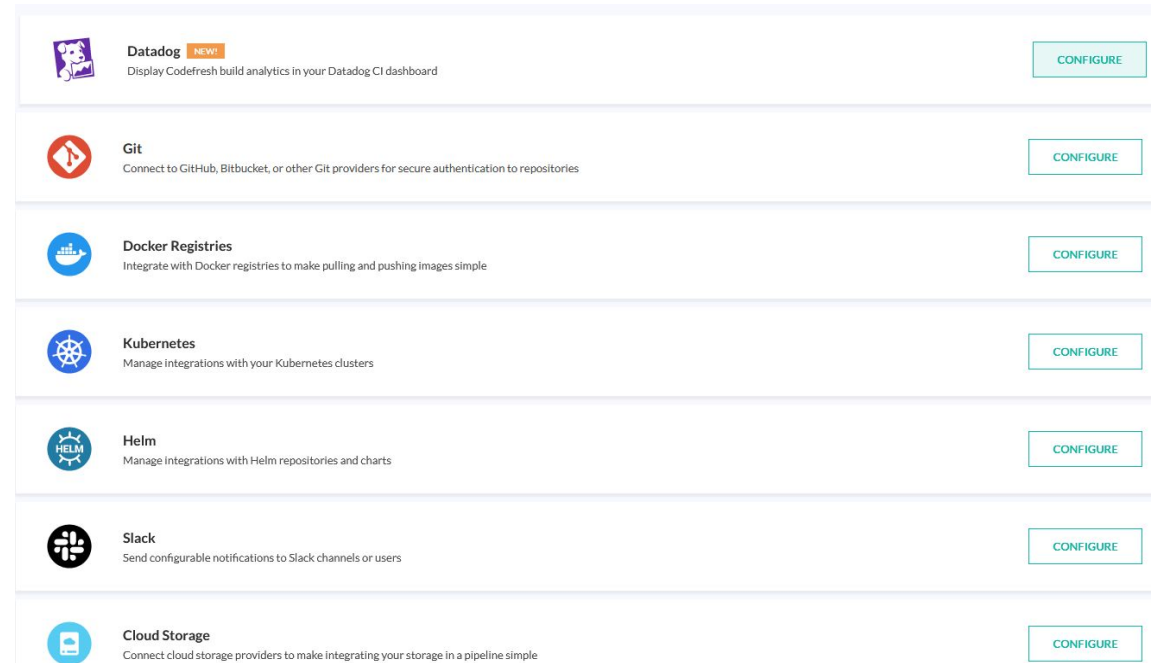
- name: deploy to cluster
  uses: kodermax/kubectl-aws-eks@master
  env:
    KUBE_CONFIG_DATA: ${ secrets.KUBE_CONFIG_DATA_STAGING }
    ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
    ECR_REPOSITORY: my-app
    IMAGE_TAG: ${ github.sha }
  with:
    args: set image deployment/$ECR_REPOSITORY $ECR_REPOSITORY=$ECR_F
```

## Codefresh

```
deploy_to_k8:
  title: deploying to cluster
  type: deploy
  kind: kubernetes
  cluster: myDemoAKSCluster
  namespace: demo
  service: my-python-app
```

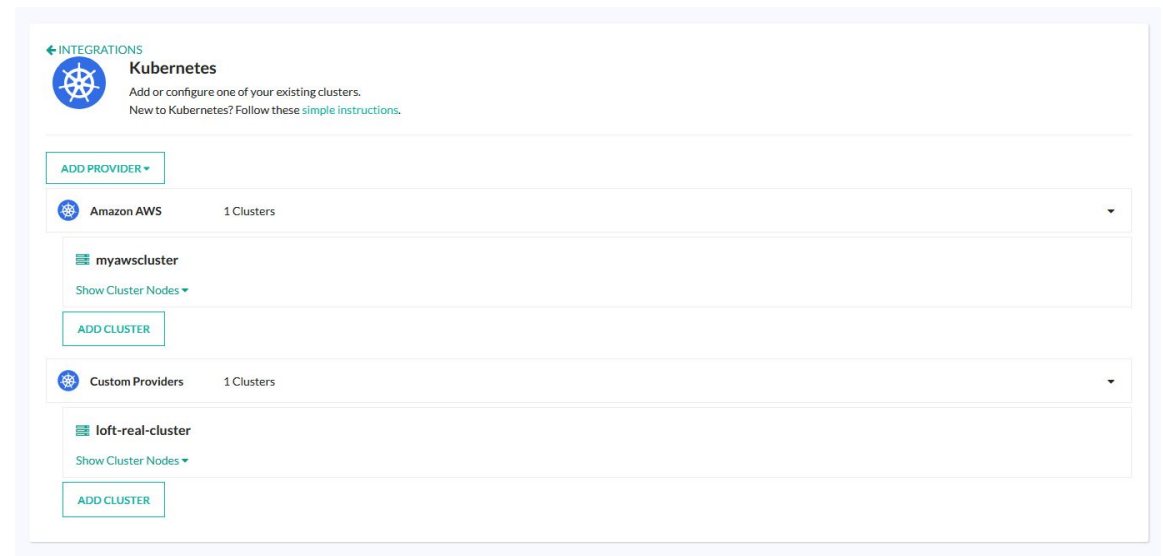
# Codefresh settings

- Attach clusters once
- Attach registries once
- Attach Helm repos once
- Attach X once (and then reference it by name)



A screenshot of the Codefresh integrations settings page. It features a list of integrations, each with an icon, a title, a brief description, and a 'CONFIGURE' button. The integrations listed are:

- Datadog** (NEW): Display Codefresh build analytics in your Datadog CI dashboard.
- Git**: Connect to GitHub, Bitbucket, or other Git providers for secure authentication to repositories.
- Docker Registries**: Integrate with Docker registries to make pulling and pushing images simple.
- Kubernetes**: Manage integrations with your Kubernetes clusters.
- Helm**: Manage integrations with Helm repositories and charts.
- Slack**: Send configurable notifications to Slack channels or users.
- Cloud Storage**: Connect cloud storage providers to make integrating your storage in a pipeline simple.



A screenshot of the Codefresh Kubernetes configuration page. The page title is 'Kubernetes' and it includes instructions: 'Add or configure one of your existing clusters. New to Kubernetes? Follow these [simple instructions](#).' Below the instructions, there are two sections for adding clusters:

- Amazon AWS**: 1 Clusters. A dropdown menu shows a cluster named 'myawscluster' with a 'Show Cluster Nodes' link and an 'ADD CLUSTER' button.
- Custom Providers**: 1 Clusters. A dropdown menu shows a cluster named 'loft-real-cluster' with a 'Show Cluster Nodes' link and an 'ADD CLUSTER' button.

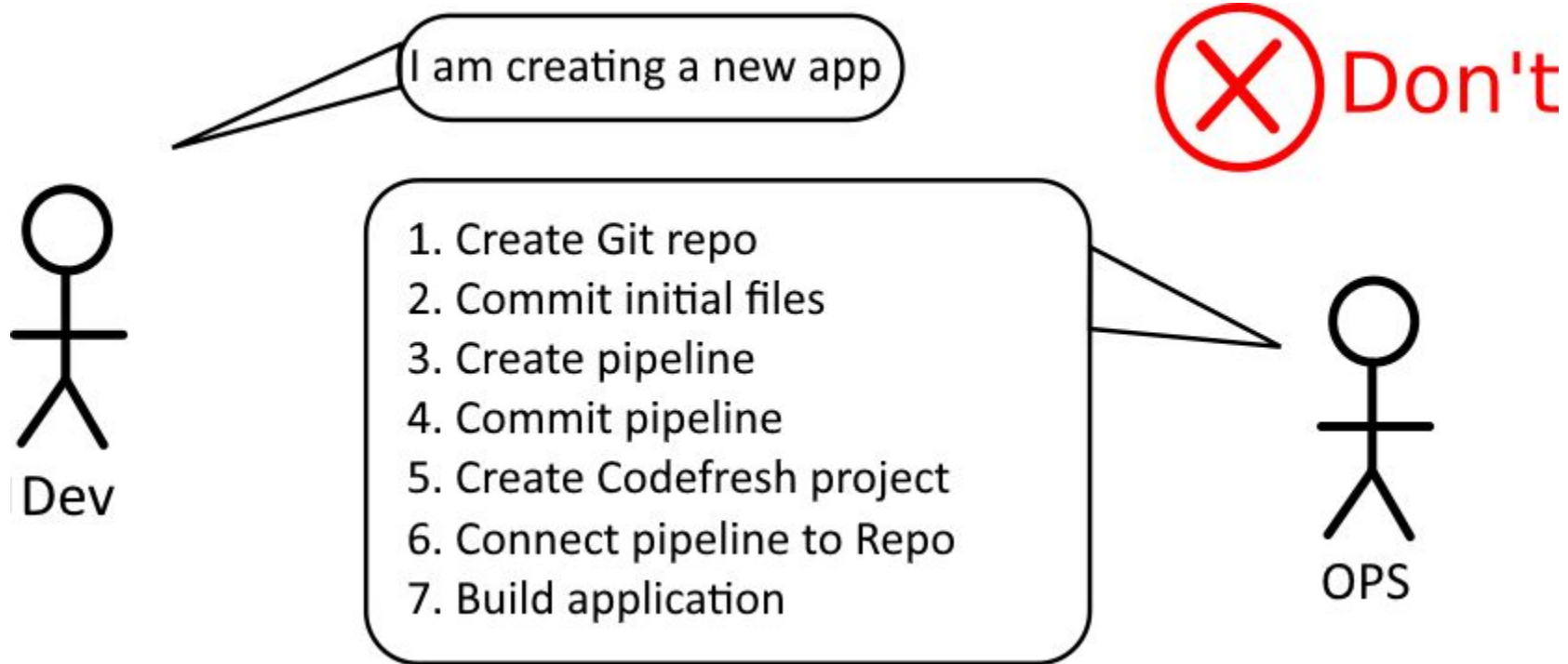
# Demo 2

**Creating a Port blueprint just  
for developers**

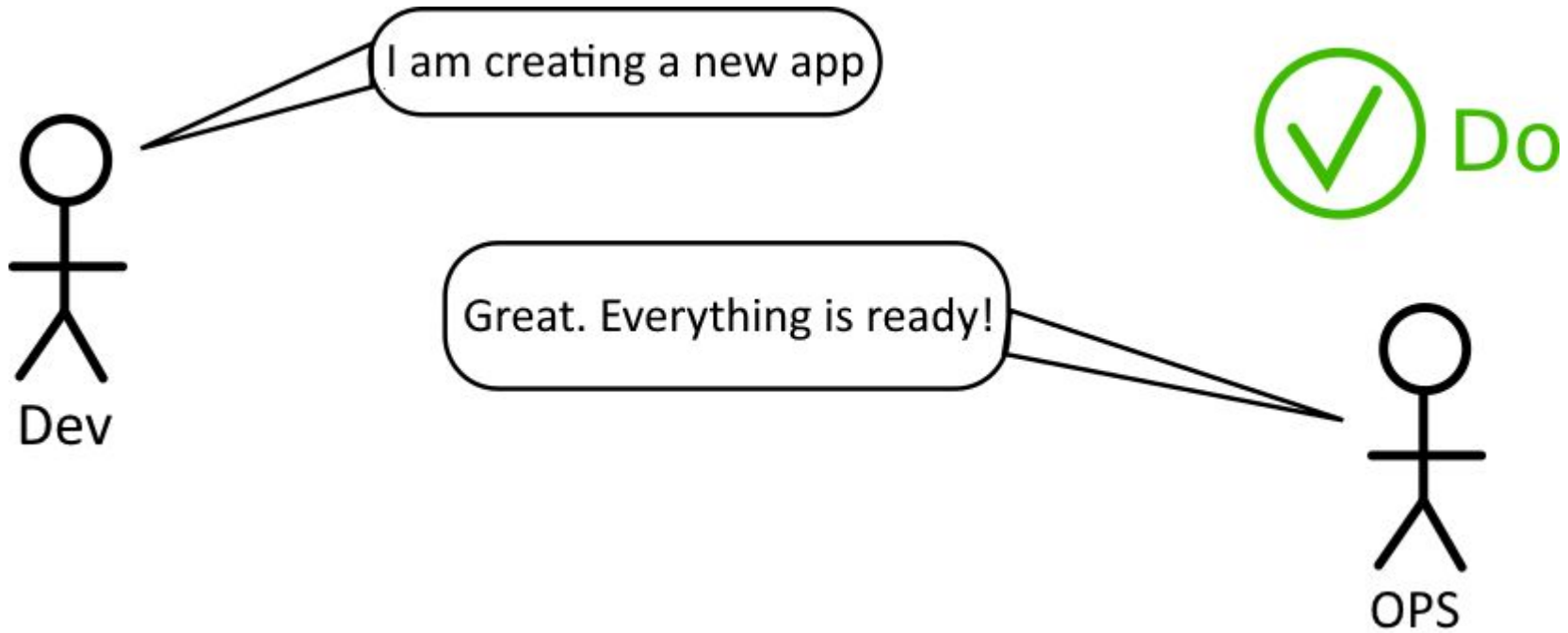
# Real World example

Brand new application/micro-service

# What usually happens



# What should happen



# Master pipeline

Create **Everything** a developer needs in a single step



## 01 Create Git Repo

Happens via GitHub  
terraform provider

## 02 Create Clusters

Happens via Vcluster chart

## 03 Create Pipelines

Happens via Codefresh  
Terraform provider

## 04 Connect Clusters

Happens via Codefresh

## 05 Ready!

Developer can start working

# Terraform everywhere

- Codefresh can run Terraform natively
- Port can inject data with Terraform
- Terraform provider for GitHub
- Terraform provider for Virtual Clusters
- Terraform provider for Codefresh itself





# Terraform provider (Codefresh)

## Example Usage

```
resource "codefresh_project" "test" {  
  name = "myproject"  
  
  tags = [  
    "production",  
    "docker",  
  ]  
  
  variables = {  
    go_version = "1.13"  
  }  
}
```

# Terraform provider (GitHub)

## Example Usage

```
resource "github_repository" "example" {  
  name          = "example"  
  description   = "My awesome codebase"  
  
  visibility    = "public"  
  
  template {  
    owner          = "github"  
    repository     = "terraform-template-module"  
    include_all_branches = true  
  }  
}
```

# Master pipeline



EVENT  
Manual

PIPELINE  
create-everything

INITIATOR  
kostis-codefresh

COMPLETED Build Completed Successfully

3 min 1 s

3 minutes ago

Small

LOG

61.93kB

EDIT PIPELINE

RESTART

⚡ Initializing Process

13 s

PREPARE →

INFRA →

CONFIGURE

PIPELINES



Cloning repository  
Step type: git-clone

2 s



Creating virtual clusters  
Step type: freestyle

1 min 31 s



Getting Virtual Contexts  
Step type: freestyle

8 s



Creating pipelines and repo  
Step type: freestyle

15 s



Choosing Real cluster  
Step type: freestyle

8 s



Listing virtual clusters  
Step type: freestyle

8 s



Creating Service accounts  
Step type: freestyle

7 s

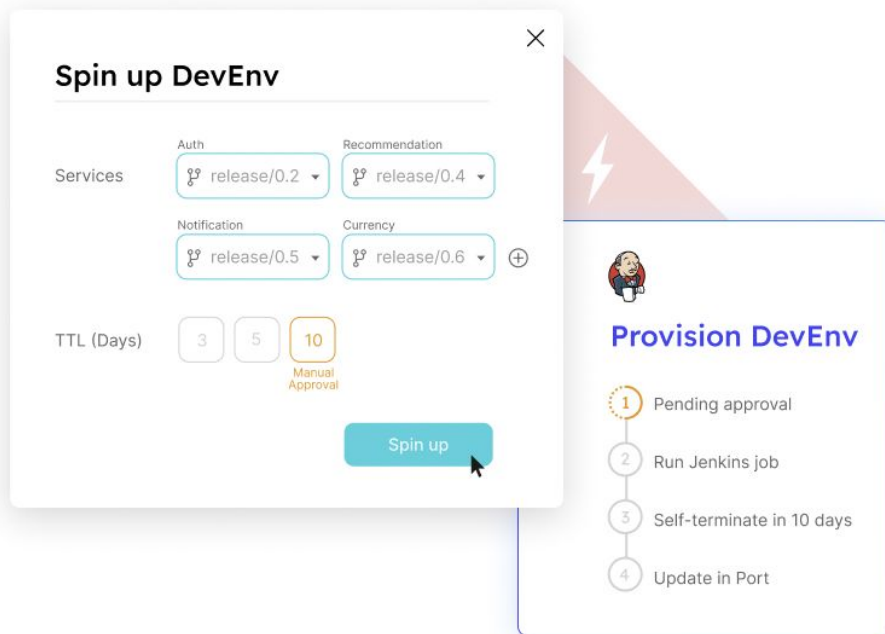


Attaching VClusters to Codefr  
Step type: freestyle

22 s

# Developer self-service

Port's internal developer portal lets developers provision, terminate and perform day-2 operations on any asset exposed in the catalog, within the policies and guardrails you've set.



**Spin up DevEnv**

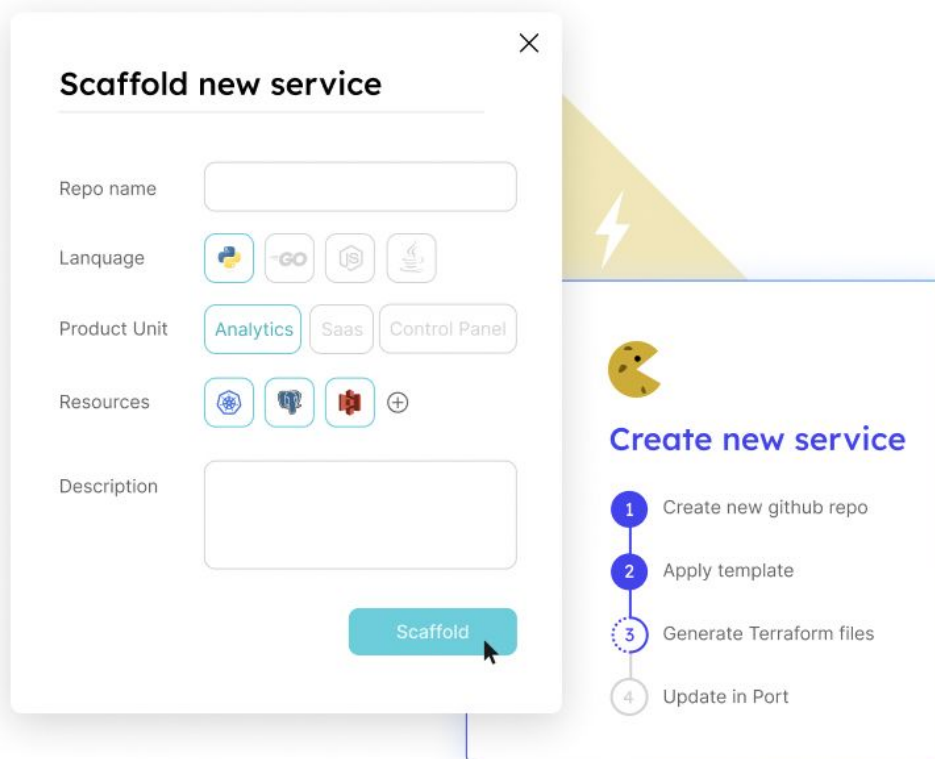
Services: Auth: release/0.2, Recommendation: release/0.4, Notification: release/0.5, Currency: release/0.6

TTL (Days): 3, 5, 10 (Manual Approval)

**Spin up**

**Provision DevEnv**

- 1 Pending approval
- 2 Run Jenkins job
- 3 Self-terminate in 10 days
- 4 Update in Port



**Scaffold new service**

Repo name: [input field]

Language: Python, GO, JS, Java

Product Unit: Analytics, Saas, Control Panel

Resources: [icons] +

Description: [input field]

**Scaffold**

**Create new service**

- 1 Create new github repo
- 2 Apply template
- 3 Generate Terraform files
- 4 Update in Port

# Demo 3

**Creating everything a  
developer needs with a  
single click**

# Recap

- Create a Port model with Codefresh/Terraform
- Create Self-service actions for devs
- Actions call Codefresh pipelines/terraform/tools
  
- Developers are happy (self-serve)
- DevOps/SREs are happy (no involvement at all)
- DexEx are happy (no blockers, no tickets)

**Thank you!** 🚀

**getport.io codefresh.io**