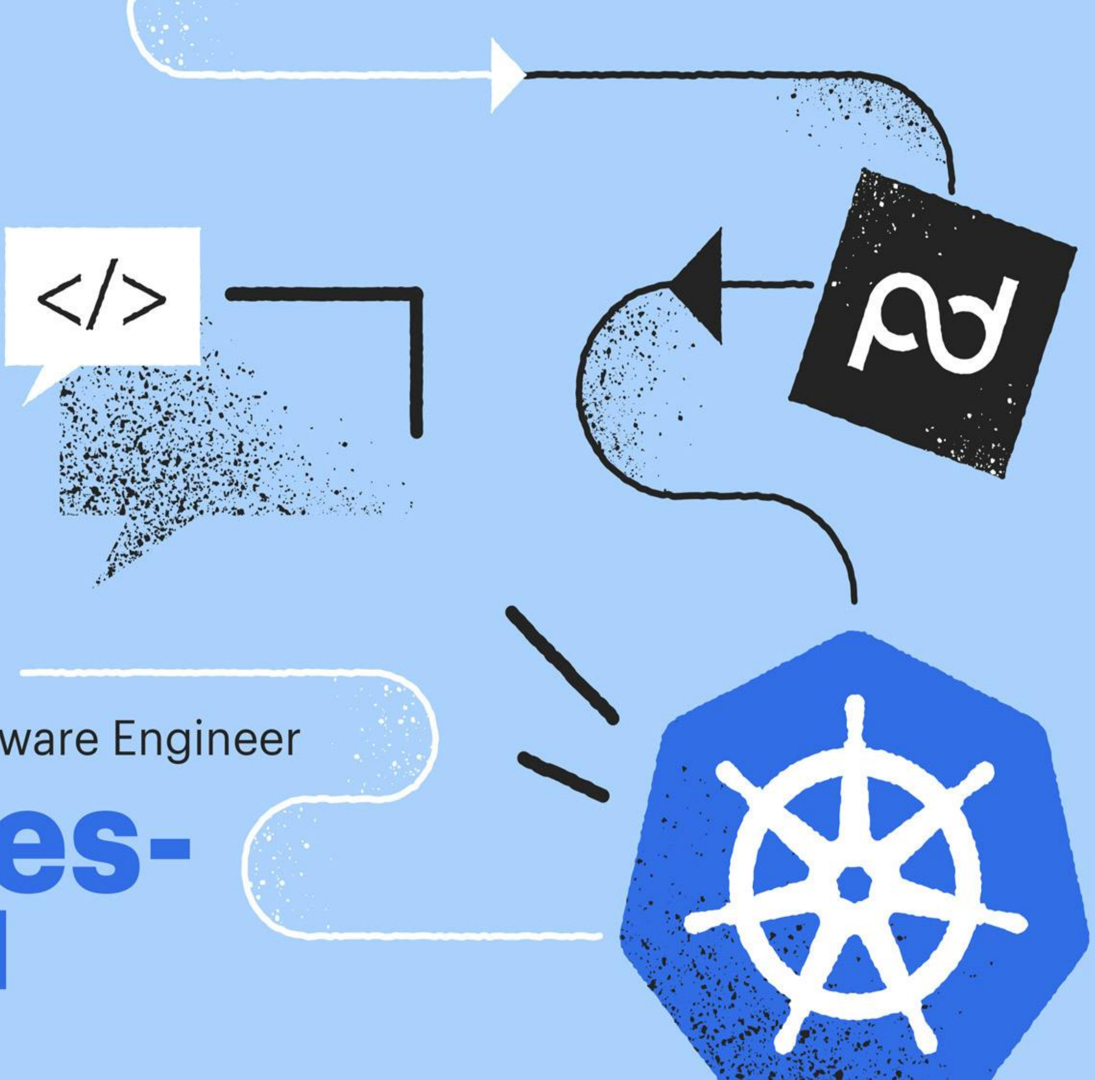
Никита Гришко / Lead Software Engineer

Kubernetes-зируемся



whoami

- Lead Software Engineer at PandaDoc
- 10 years experience in backend development
- Moved from product development to platform engineering

A few years ago...

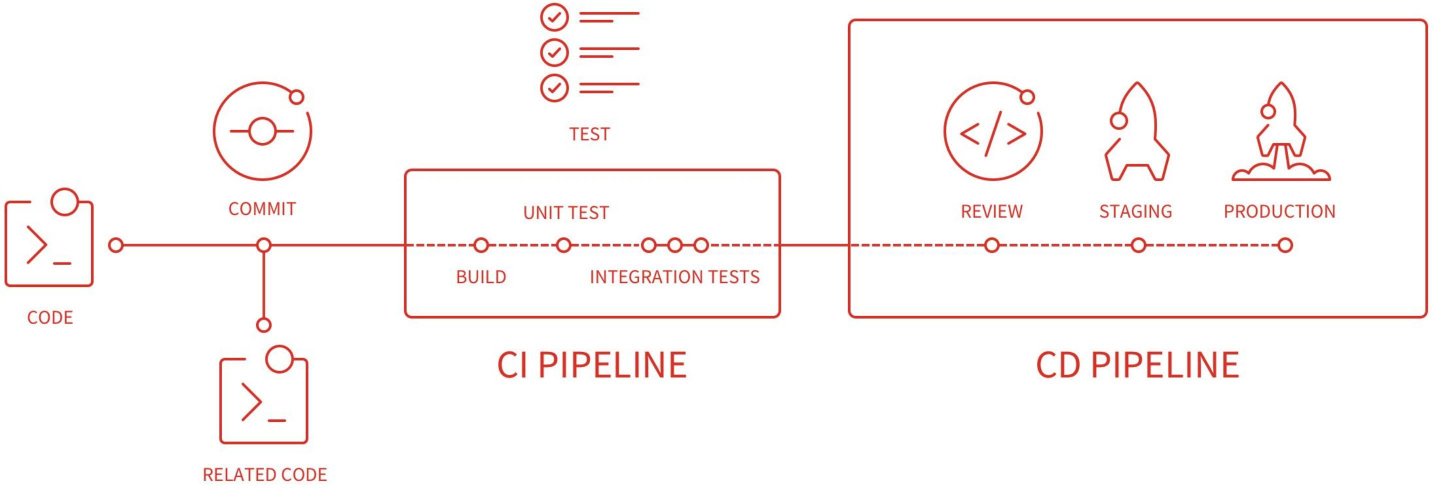
https://www.youtube.com/watch?v=QhdggHKnVtk

A few years ago...

- AWS EC2 + Docker
- Gitlab for source code management and CI
- First steps in continuous integration

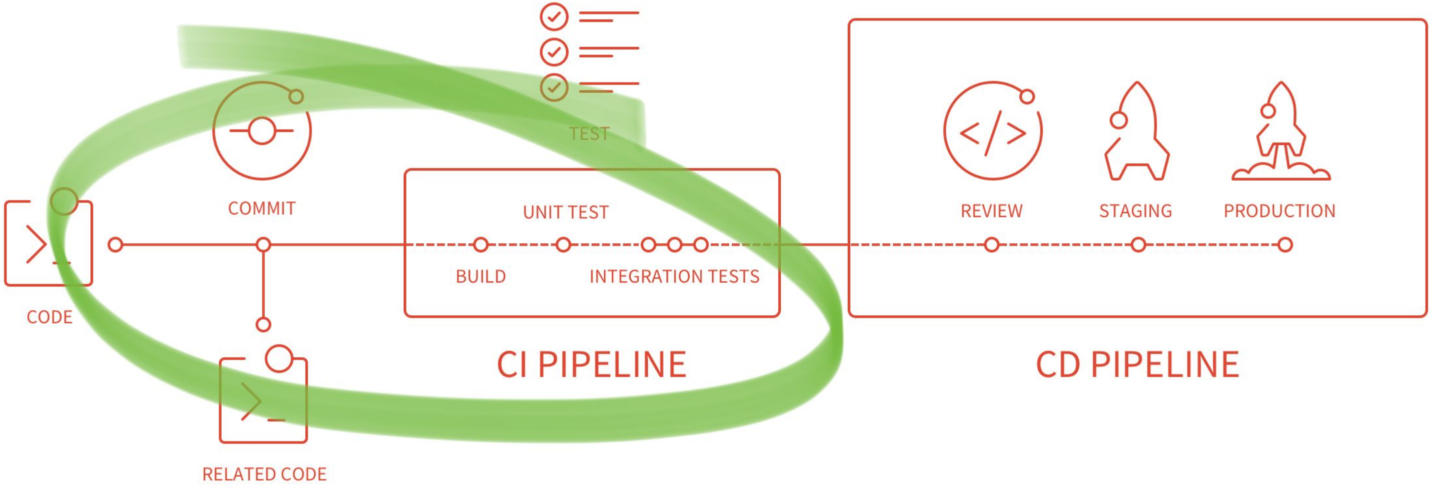
https://www.youtube.com/watch?v=QhdggHKnVtk

A few years ago...

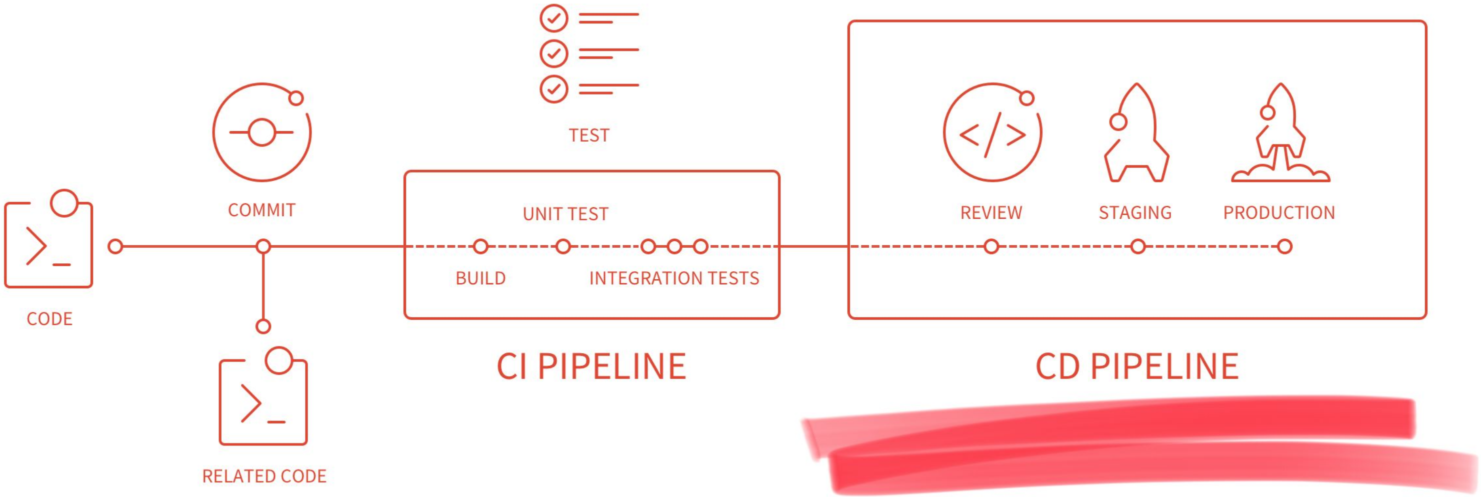


PandaDoc

A few years ago...



A few years ago...



No, no, no, no

- No bu****it bingo
- No manuals retelling
- No “silver bullet” solutions
- No complex technical solutions

Yes!

- Kubernetes? Why?
- From AWS EC2 to AWS EKS: our step by step guide
- Our CI/CD pipeline

Still not interested yet?

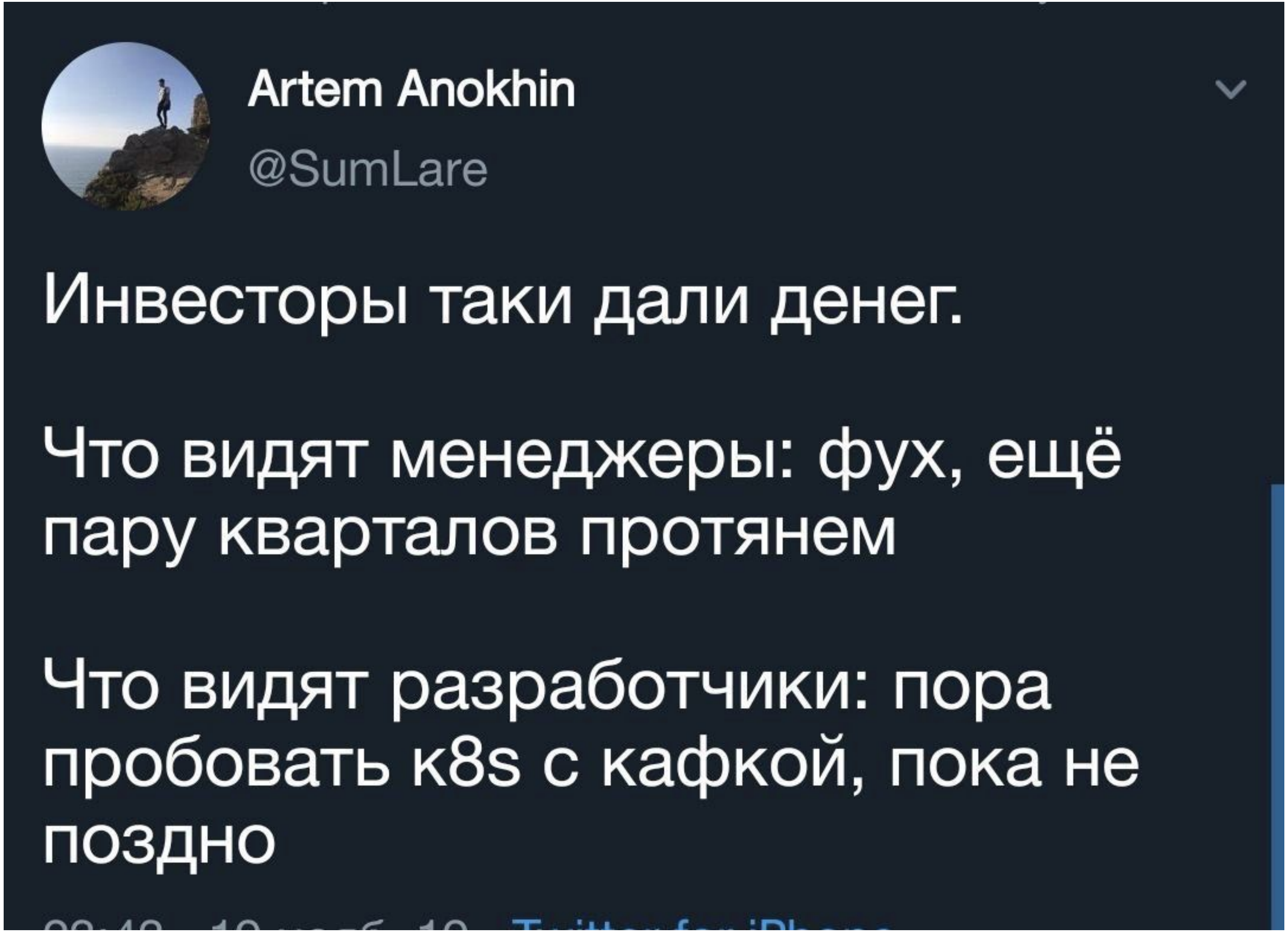
- More than 50 services migrated in 5 months (including legacy ones)
- Over 20+ environments run on Kubernetes
- No epic fails, please believe me

Still not interested yet?

- More than 50 services migrated in 5 months (including **legacy** ones)

Using Kubernetes for Legacy services.

Kubernetes? Why?



Artem Anokhin

@SumLare

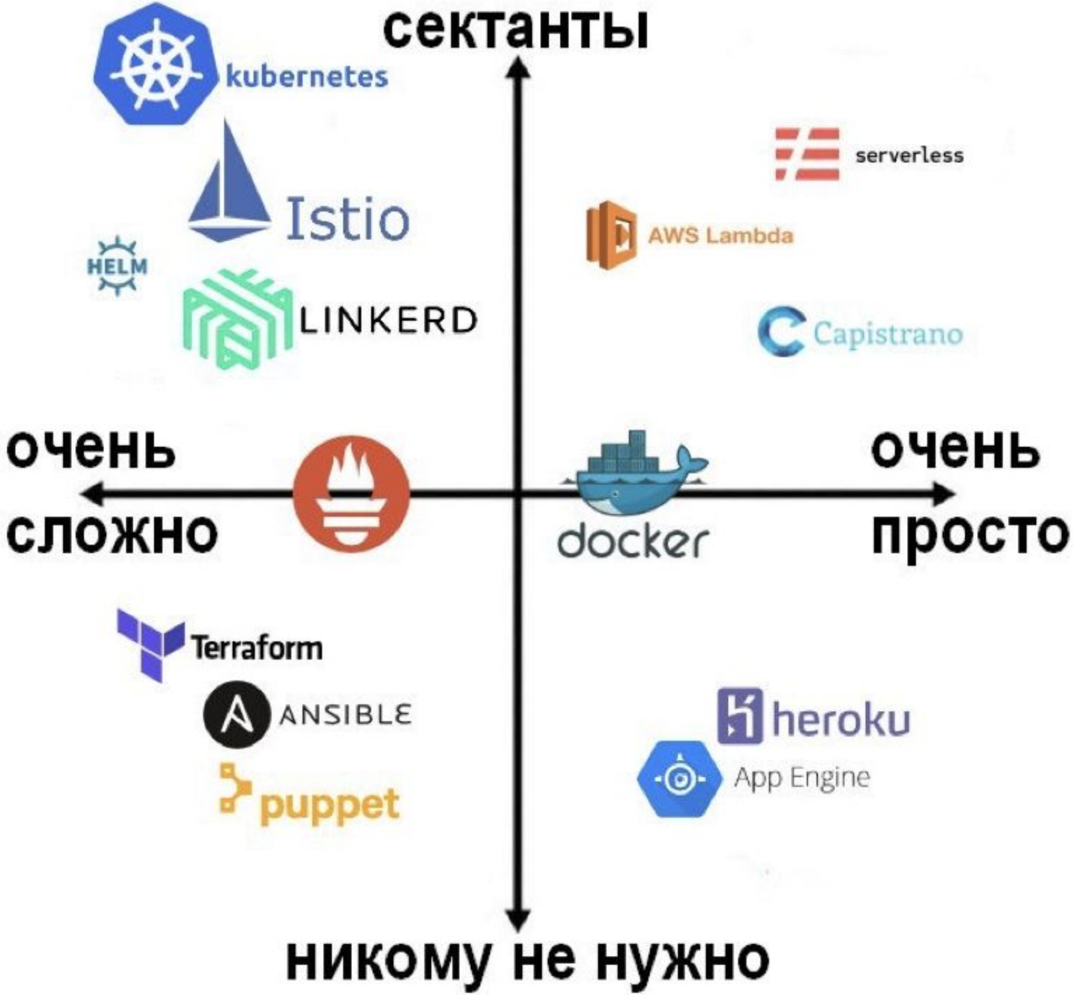
Инвесторы таки дали денег.

Что видят менеджеры: фух, ещё пару кварталов протянем

Что видят разработчики: пора пробовать k8s с кафкой, пока не поздно

Kubernetes? Why?

- Terraform + Ansible


сектанты
очень сложно
очень просто
никому не нужно
kubernetes
Istio
HELM
LINKERD
serverless
AWS Lambda
Capistrano
docker
Terraform
ANSIBLE
puppet
heroku
App Engine

Kubernetes? Why?

- Terraform + Ansible:
    - ~~Too difficult and no one wants it~~
    - A lot of repositories to put changes
    - Hard and painful rollback
    - Anyone from another team can accidentally break your deployment configuration
    - 2-4 days for initial service setup
    - **Bad cycle time experience**

Kubernetes? Why?

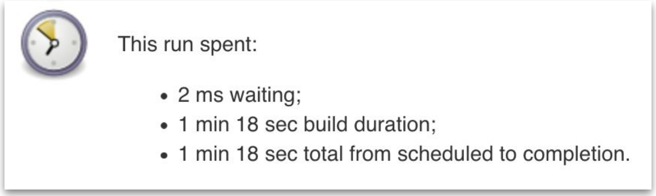
- Terraform is slow…
    - Unexpected load on your service? Want to increase nodes count or change node type?
    - **Again bad cycle time experience**

Kubernetes? Why?

- Ansible is slow too…
  - Install system packages
  - Setup users
  - Render templates
  - Pull and run containers
  - **And again bad cycle time experience**

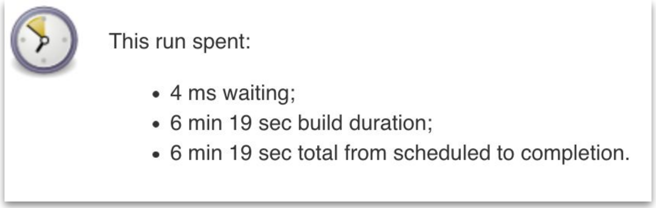
Kubernetes? Why?

- Ansible is slow too…



This run spent:

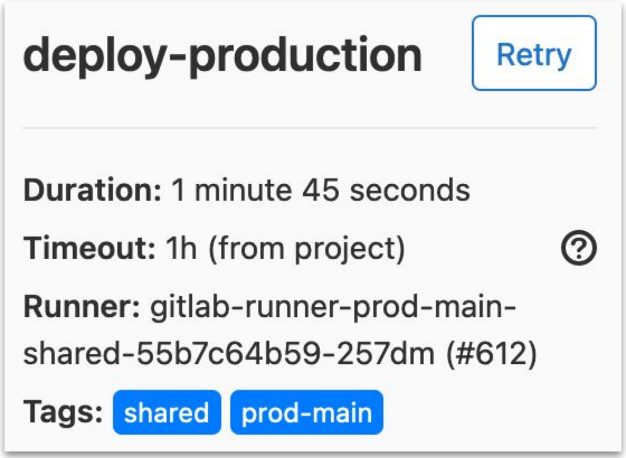
- 2 ms waiting;
- 1 min 18 sec build duration;
- 1 min 18 sec total from scheduled to completion.



This run spent:

- 4 ms waiting;
- 6 min 19 sec build duration;
- 6 min 19 sec total from scheduled to completion.

VS



deploy-production

Duration: 1 minute 45 seconds

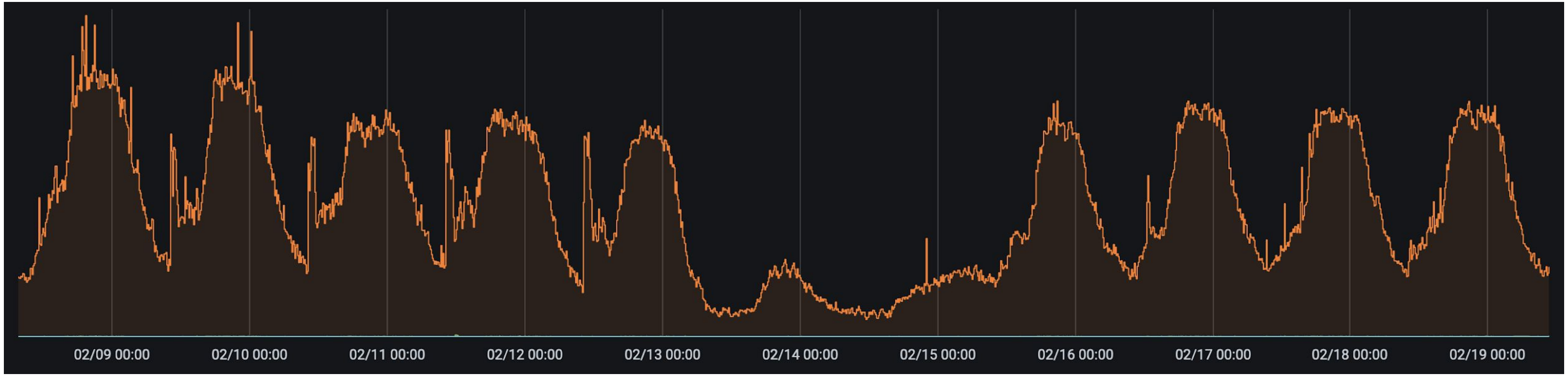
Timeout: 1h (from project)

Runner: gitlab-runner-prod-main-shared-55b7c64b59-257dm (#612)

Tags: shared prod-main

PandaDoc

Kubernetes? Why?



Time-dependent load on services

Kubernetes? Why?

- Time-dependent load on services
- Bad resources utilization
- Risk of downtime in case of unexpected load

So why? What we want?

- Everything in one repository: source code, deployment configuration, even dashboards and alerts
- (Auto)scaling and resources optimization
- Cost optimization
- **Better cycle time and development experience**

Maybe XYZ?

- Kubernetes de facto industry standard
- A lot of experience and tools in community

ARE Y'ALL WITH THE CULT
WE'RE NOT A CULT
WE BELIEVE YOU CAN DEFINE ANY COMPLEX DEPLOYMENT USING SOLELY YAML AND GITOPS
YEAH, THIS IS IT

Ready for maintenance standard

Ready for maintenance standard?

- Only Docker images
- Scripts: migrate.sh and entrypoint.sh
- Configuration using environment variables
- Logs to STDOUT in JSON
- Secrets in HashiCorp Vault
- Expose 4284 port for health check
- Prometheus and standardized set of metrics
- Transports: NATS and Kafka
- ...

Ready for maintenance standard?

- gw* (gwjava-*, gwpy-*)
- ms* (msjava-*, mspy-*, msscala-*)

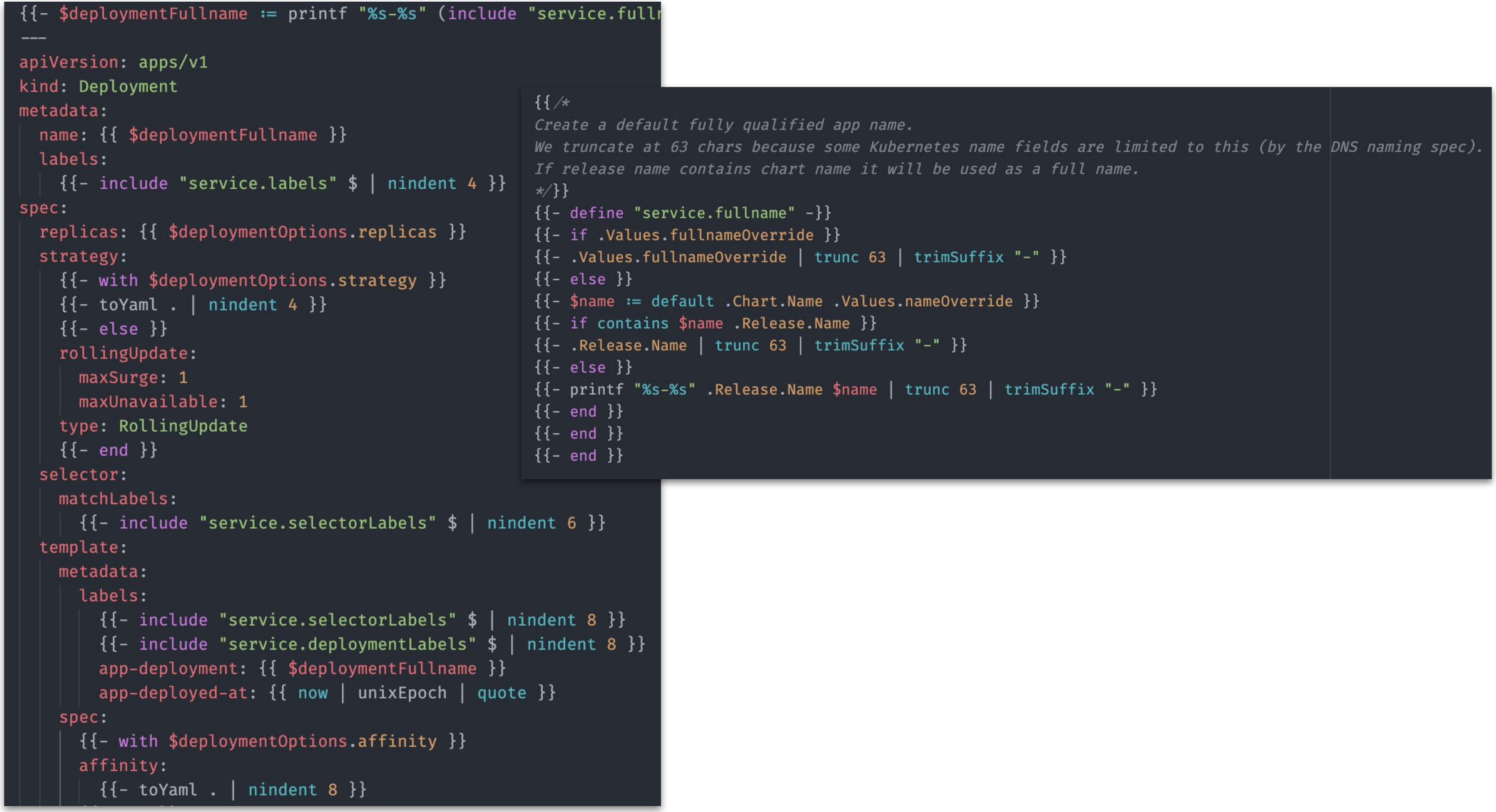
Ready for maintenance standard!

Standard allows us to unify services and simplify deployments

HELM

Helm charts

- Designed by aliens for predators...

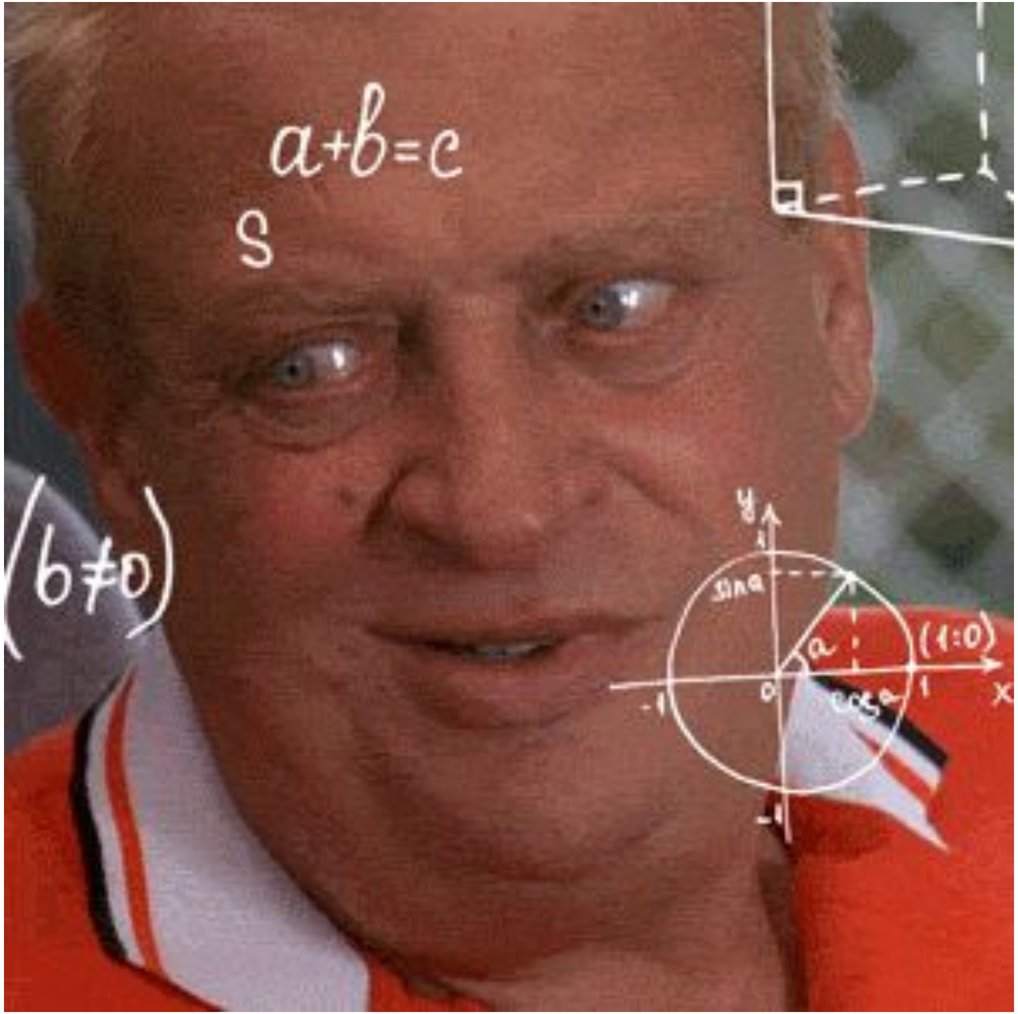


```
{{- $deploymentFullname := printf "%s-%s" (include "service.full
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ $deploymentFullname }}
  labels:
    {{- include "service.labels" $ | nindent 4 }}
spec:
  replicas: {{ $deploymentOptions.replicas }}
  strategy:
    {{- with $deploymentOptions.strategy }}
    {{- toYaml . | nindent 4 }}
    {{- else }}
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
    {{- end }}
  selector:
    matchLabels:
      {{- include "service.selectorLabels" $ | nindent 6 }}
  template:
    metadata:
      labels:
        {{- include "service.selectorLabels" $ | nindent 8 }}
        {{- include "service.deploymentLabels" $ | nindent 8 }}
        app-deployment: {{ $deploymentFullname }}
        app-deployed-at: {{ now | unixEpoch | quote }}
    spec:
      {{- with $deploymentOptions.affinity }}
      affinity:
        {{- toYaml . | nindent 8 }}
```

```
{{/*
Create a default fully qualified app name.
We truncate at 63 chars because some Kubernetes name fields are limited to this (by the DNS naming spec).
If release name contains chart name it will be used as a full name.
*/}}
{{- define "service.fullname" -}}
{{- if .Values.fullnameOverride }}
{{- .Values.fullnameOverride | trunc 63 | trimSuffix "-" }}
{{- else }}
{{- $name := default .Chart.Name .Values.nameOverride }}
{{- if contains $name .Release.Name }}
{{- .Release.Name | trunc 63 | trimSuffix "-" }}
{{- else }}
{{- printf "%s-%s" .Release.Name $name | trunc 63 | trimSuffix "-" }}
{{- end }}
{{- end }}
{{- end }}
```

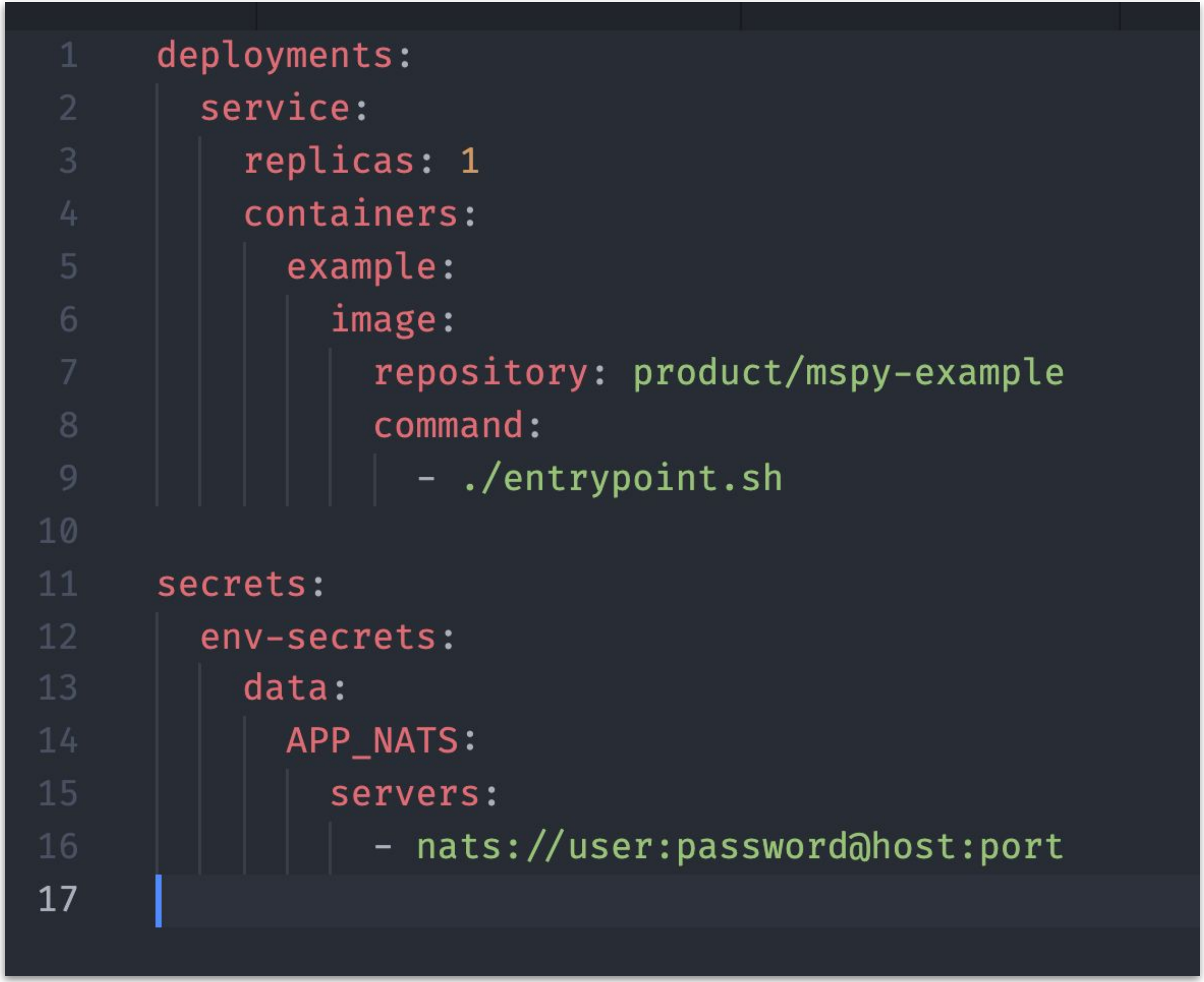
Helm charts

- Designed by aliens for predators…
- YAML + Golang template engine
- Too complex and too easy to make a mistake
- **Can we provide a better experience?**


a+b=c
S
(b≠0)
y
1
sina
(1:0)
a
0
1
-1
cosa
x
-1

Helm charts

- Shared Helm chart for the whole company
- Expose Helm values only for engineers
  helm/values/{defaults|preprod|prod}.yaml



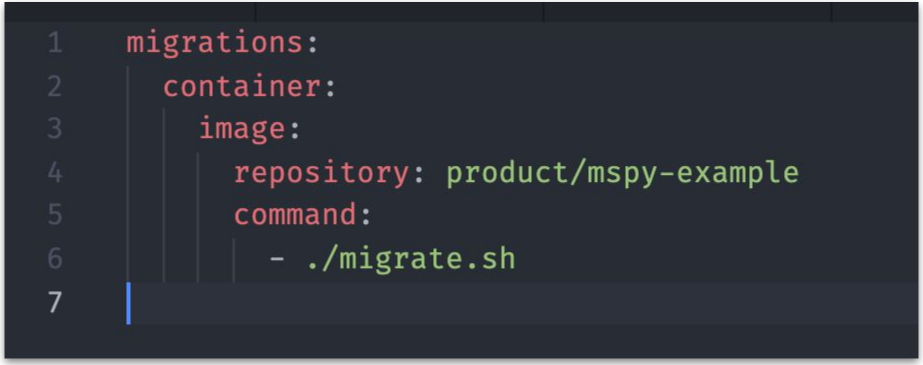
```yaml
deployments:
  service:
    replicas: 1
    containers:
      example:
        image:
          repository: product/mspy-example
          command:
            - ./entrypoint.sh

secrets:
  env-secrets:
    data:
      APP_NATS:
        servers:
          - nats://user:password@host:port
```

PandaDoc chart

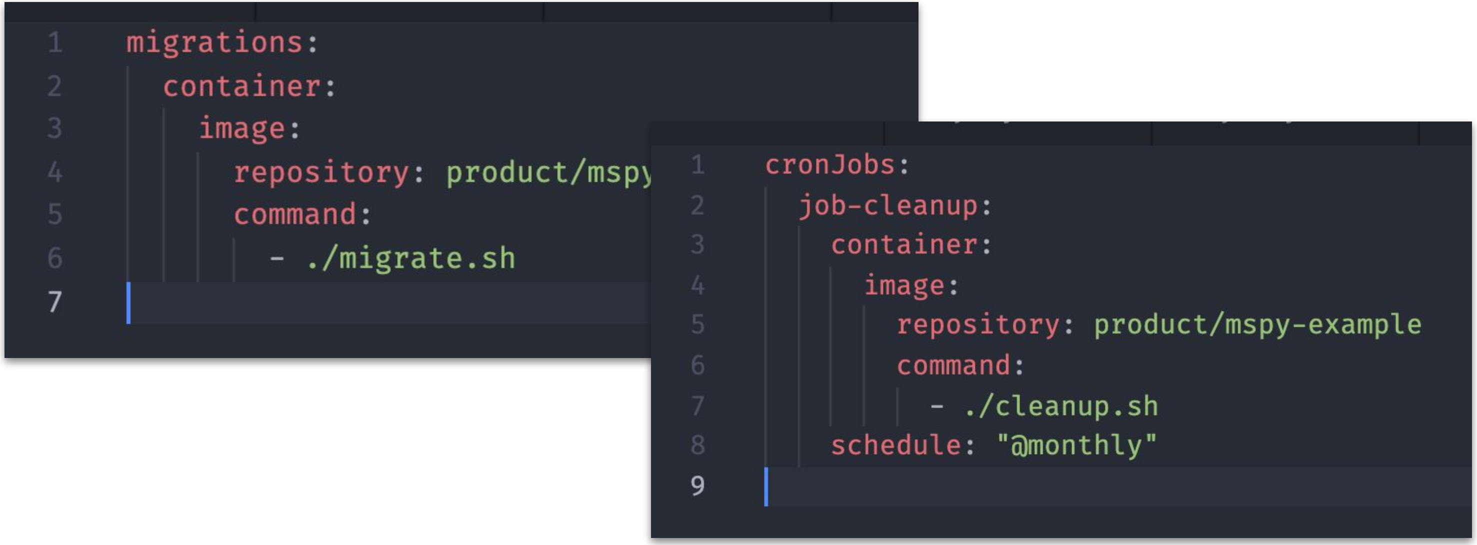
- Readiness/liveness probes
- CPU/Memory resources
- Ports
- Network services
- Service monitors
- ...

PandaDoc chart

Thanks to **ready for maintenance standard** we know everything about our services and we can provide **useful defaults**

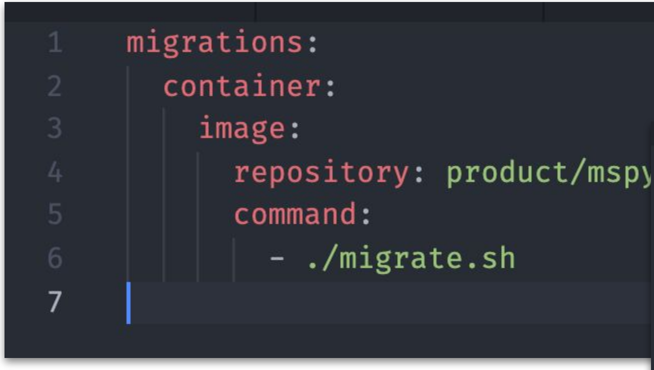


```yaml
migrations:
  container:
    image:
      repository: product/mspy-example
      command:
        - ./migrate.sh

```

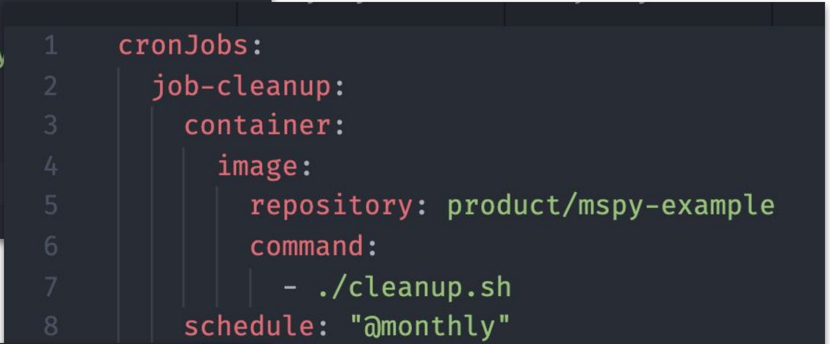


```yaml
migrations:
  container:
    image:
      repository: product/mspy
      command:
        - ./migrate.sh

```

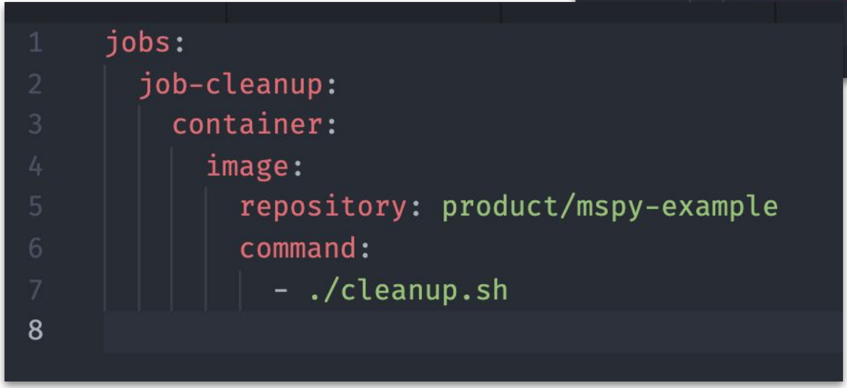
```yaml
cronJobs:
  job-cleanup:
    container:
      image:
        repository: product/mspy-example
        command:
          - ./cleanup.sh
    schedule: "@monthly"

```



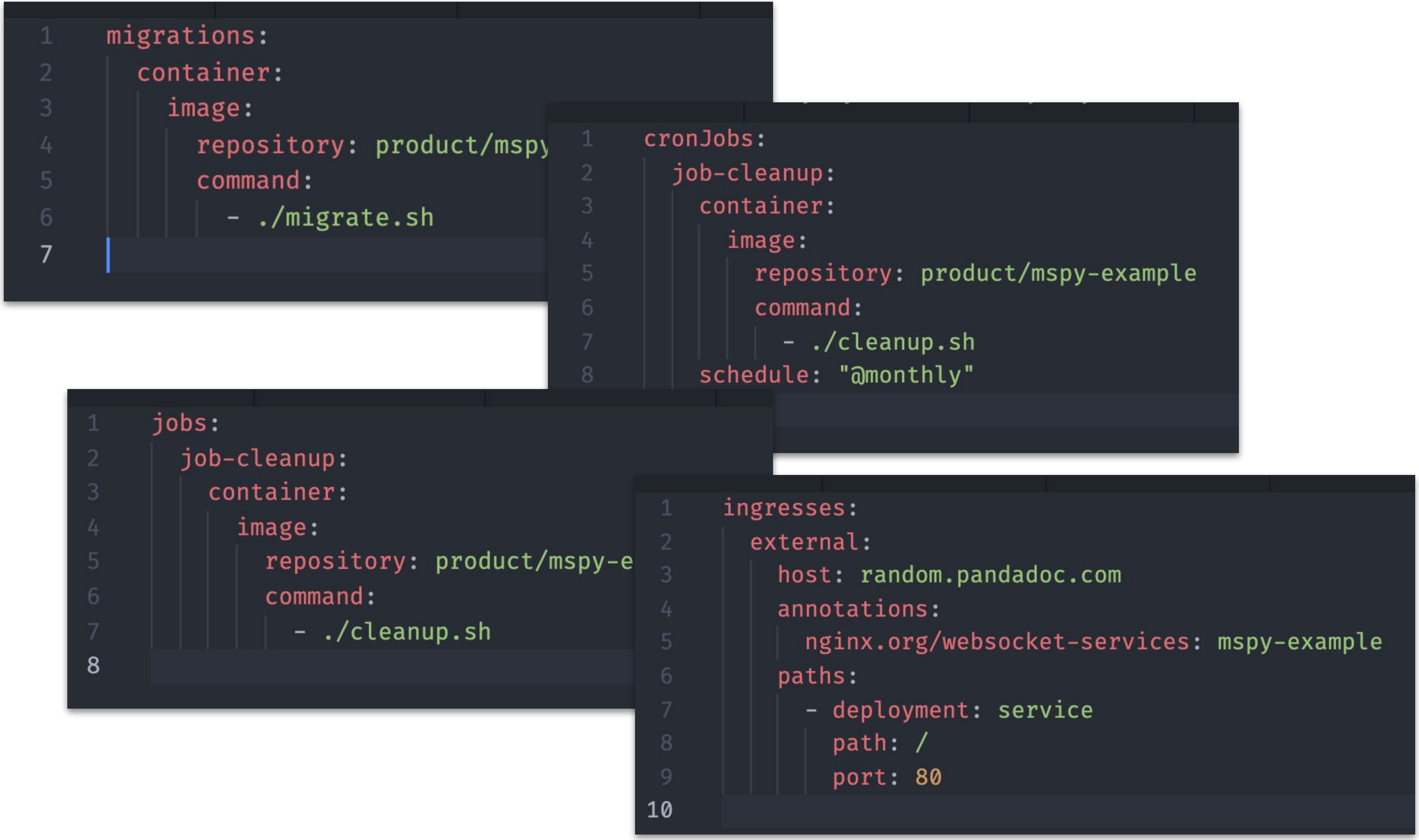
```yaml
migrations:
  container:
    image:
      repository: product/mspy
      command:
        - ./migrate.sh

```



```yaml
cronJobs:
  job-cleanup:
    container:
      image:
        repository: product/mspy-example
        command:
          - ./cleanup.sh
    schedule: "@monthly"
```



```yaml
jobs:
  job-cleanup:
    container:
      image:
        repository: product/mspy-example
        command:
          - ./cleanup.sh

```



```yaml
migrations:
  container:
    image:
      repository: product/mspy
      command:
        - ./migrate.sh
```

```yaml
cronJobs:
  job-cleanup:
    container:
      image:
        repository: product/mspy-example
        command:
          - ./cleanup.sh
    schedule: "@monthly"
```

```yaml
jobs:
  job-cleanup:
    container:
      image:
        repository: product/mspy-e
        command:
          - ./cleanup.sh
```

```yaml
ingresses:
  external:
    host: random.pandadoc.com
    annotations:
      nginx.org/websocket-services: mspy-example
    paths:
      - deployment: service
        path: /
        port: 80
```

GitLab

Gitlab CI/CD

- Build Docker image
- Run Helm (upgrade, history, ...)
- Verify that all pods are ready
- Send notifications (Slack, ...)
- Annotate Grafana dashboards
- ...

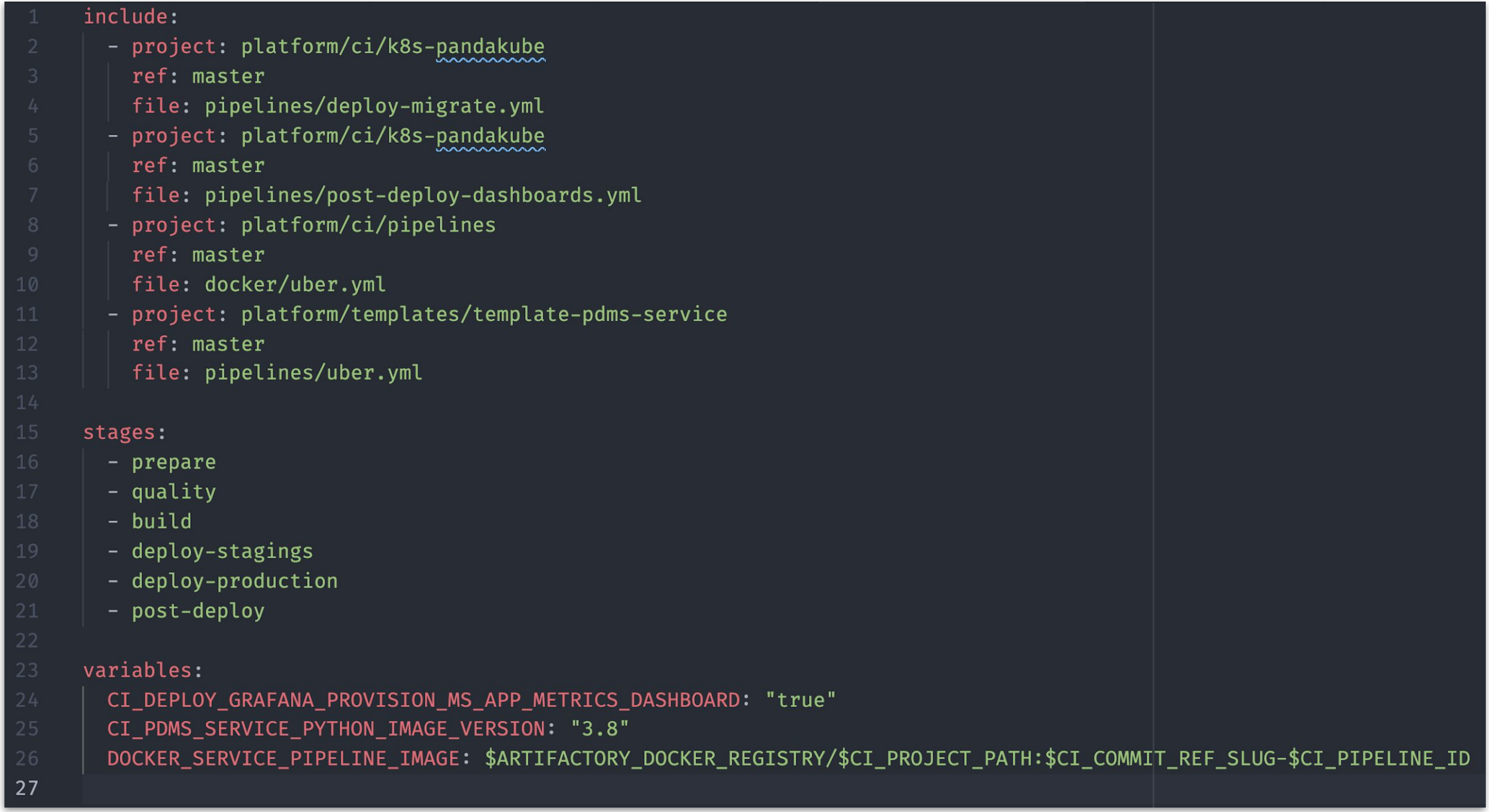
Gitlab CI/CD

- Ctrl-C / Ctrl-V is not an option
- How can we control versions of tools we use?

?
) = 1 - 1
lim (1 + a/n)^n
cosx =

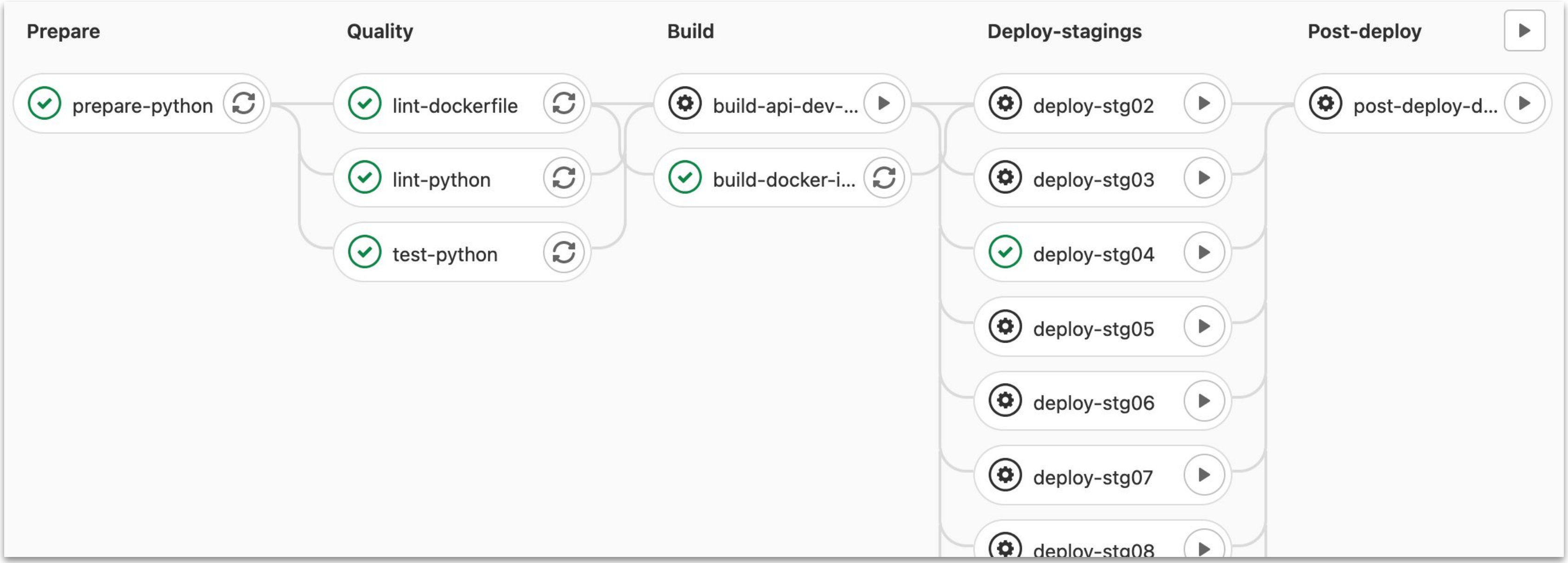
Gitlab CI/CD

- https://docs.gitlab.com/ee/ci/yaml/README.html#include
- You can use *include* to include external YAML files in your CI/CD configuration

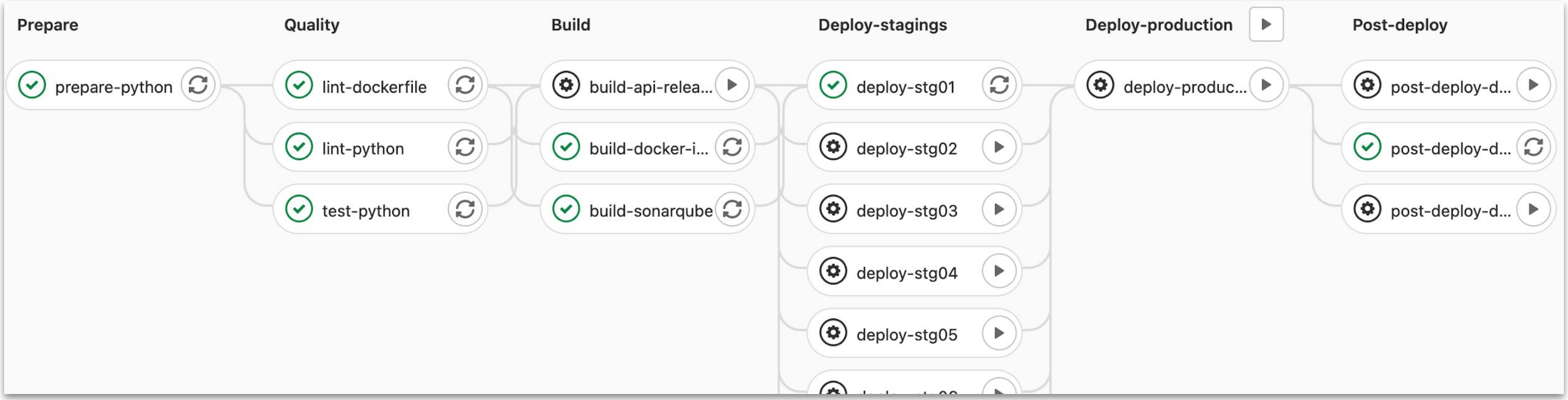


```yaml
include:
  - project: platform/ci/k8s-pandakube
    ref: master
    file: pipelines/deploy-migrate.yml
  - project: platform/ci/k8s-pandakube
    ref: master
    file: pipelines/post-deploy-dashboards.yml
  - project: platform/ci/pipelines
    ref: master
    file: docker/uber.yml
  - project: platform/templates/template-pdms-service
    ref: master
    file: pipelines/uber.yml

stages:
  - prepare
  - quality
  - build
  - deploy-stagings
  - deploy-production
  - post-deploy

variables:
  CI_DEPLOY_GRAFANA_PROVISION_MS_APP_METRICS_DASHBOARD: "true"
  CI_PDMS_SERVICE_PYTHON_IMAGE_VERSION: "3.8"
  DOCKER_SERVICE_PIPELINE_IMAGE: $ARTIFACTORY_DOCKER_REGISTRY/$CI_PROJECT_PATH:$CI_COMMIT_REF_SLUG-$CI_PIPELINE_ID
```

Gitlab CI/CD (feature branch)

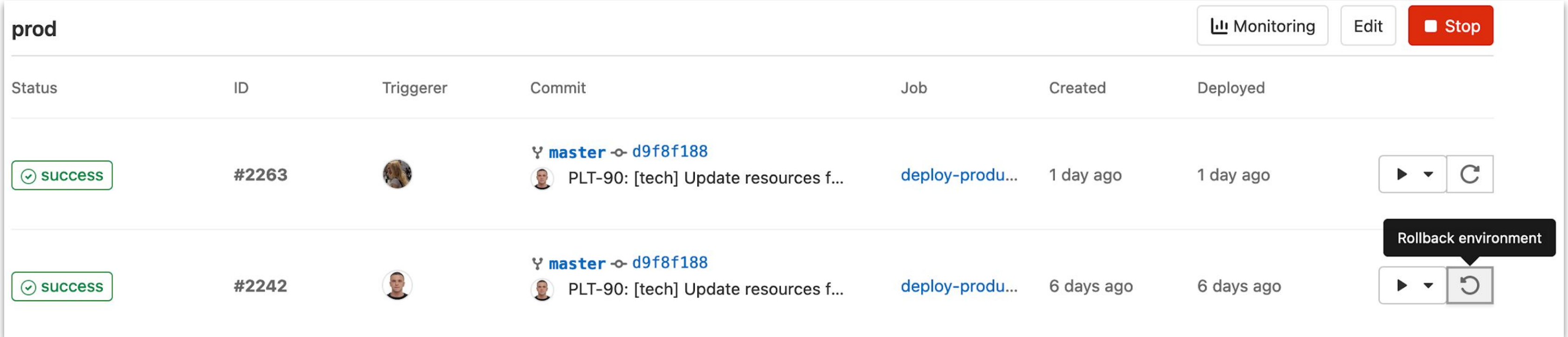


Gitlab CI/CD (master branch)



Prepare	Quality	Build	Deploy-stagings	Deploy-production ▶	Post-deploy
✓ prepare-python	✓ lint-dockerfile	⚙ build-api-relea... ▶	✓ deploy-stg01	⚙ deploy-produc... ▶	⚙ post-deploy-d... ▶
	✓ lint-python	✓ build-docker-i...	⚙ deploy-stg02 ▶		✓ post-deploy-d...
	✓ test-python	✓ build-sonarqube	⚙ deploy-stg03 ▶		⚙ post-deploy-d... ▶
			⚙ deploy-stg04 ▶		
			⚙ deploy-stg05 ▶		

Gitlab CI/CD (rollbacks)



prod

Status	ID	Triggerer	Commit	Job	Created	Deployed
success	#2263		master d9f8f188 PLT-90: [tech] Update resources f...	deploy-produ...	1 day ago	1 day ago
success	#2242		master d9f8f188 PLT-90: [tech] Update resources f...	deploy-produ...	6 days ago	6 days ago

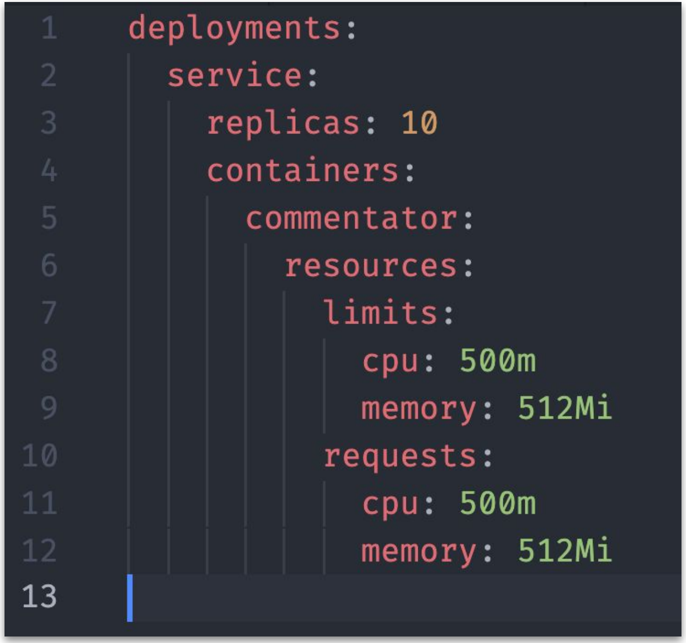
Gitlab CI/CD

- k8s-pandakube — our golden image for Kubernetes deployments
- Pinned versions of Helm and kubectl
- Helm charts
- Grafana dashboards
- ...

(Auto)scaling

(Manual)scaling

Manual scaling is easy as possible and can be applied in a few minutes (depends on the time of pipeline)



```yaml
deployments:
  service:
    replicas: 10
    containers:
      commentator:
        resources:
          limits:
            cpu: 500m
            memory: 512Mi
          requests:
            cpu: 500m
            memory: 512Mi

```

(Auto)scaling

- Horizontal Pod Autoscaler
- https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
- CPU / Memory

(Auto)scaling

- Horizontal Pod Autoscaler
- https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
- CPU / Memory
- **We want to scale by Prometheus metrics**

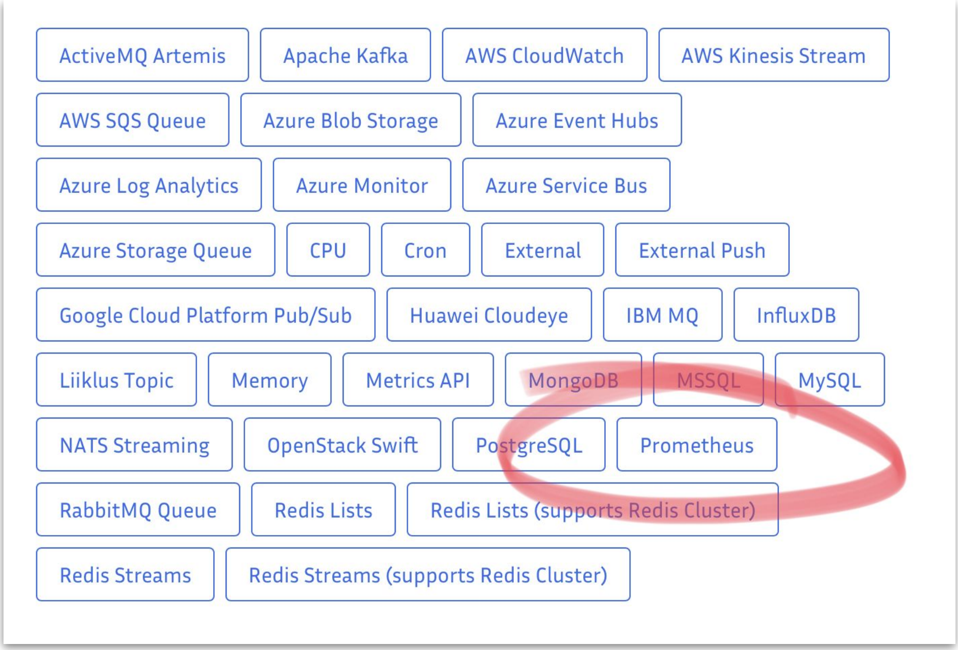
(Auto)scaling

- KEDA — Kubernetes Event-driven Autoscaling
- https://keda.sh/
- KEDA works alongside standard Kubernetes components like the Horizontal Pod Autoscaler and can extend functionality without overwriting or duplication

PandaDoc

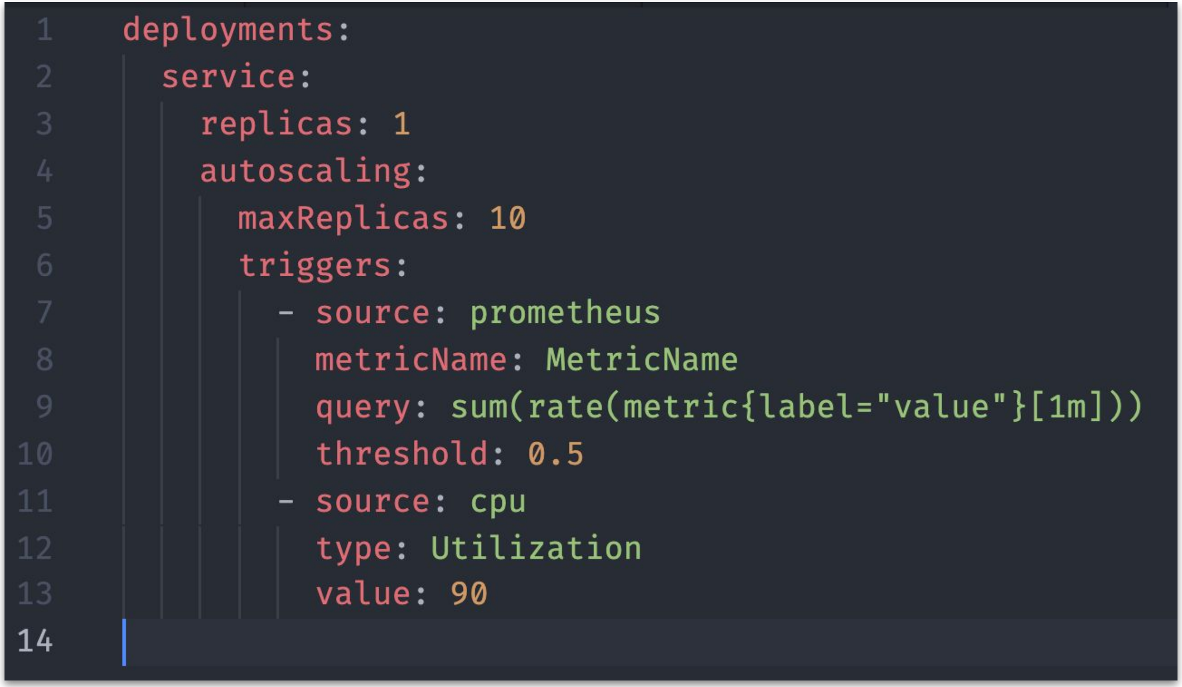
(Auto)scaling

Available scalers for KEDA 2.2



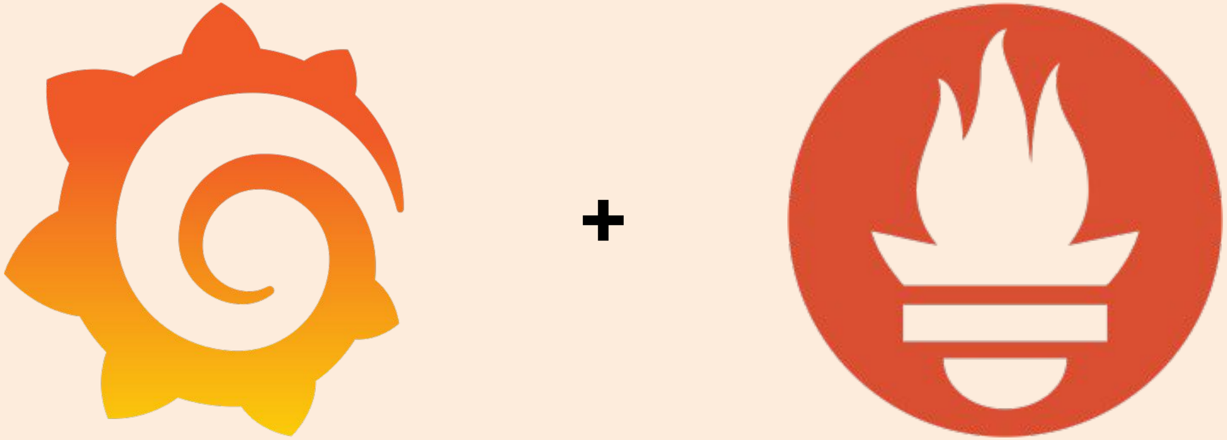
ActiveMQ Artemis
Apache Kafka
AWS CloudWatch
AWS Kinesis Stream
AWS SQS Queue
Azure Blob Storage
Azure Event Hubs
Azure Log Analytics
Azure Monitor
Azure Service Bus
Azure Storage Queue
CPU
Cron
External
External Push
Google Cloud Platform Pub/Sub
Huawei Cloudeye
IBM MQ
InfluxDB
Liiklus Topic
Memory
Metrics API
MongoDB
MSSQL
MySQL
NATS Streaming
OpenStack Swift
PostgreSQL
Prometheus
RabbitMQ Queue
Redis Lists
Redis Lists (supports Redis Cluster)
Redis Streams
Redis Streams (supports Redis Cluster)

(Auto)scaling



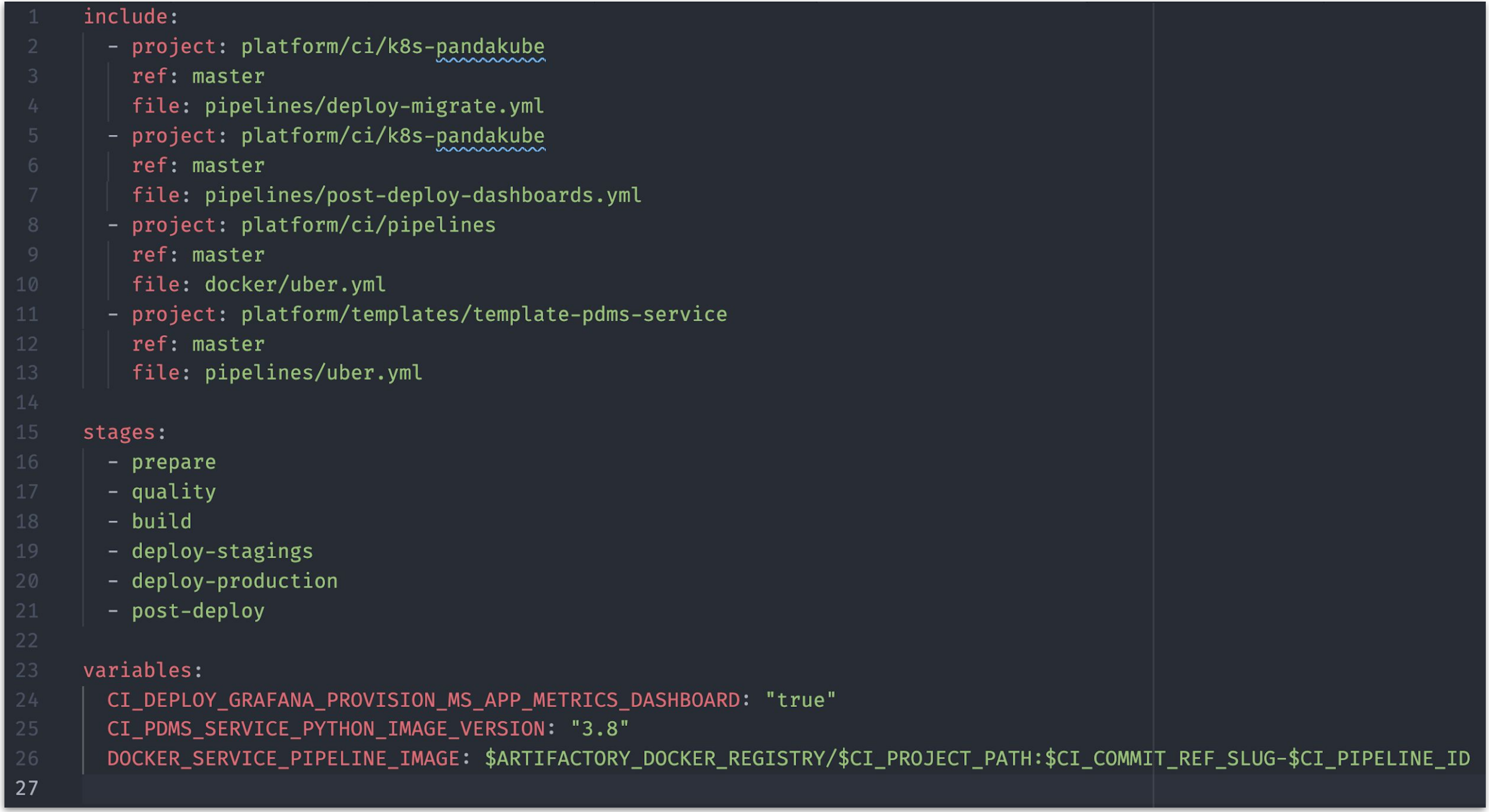
```yaml
deployments:
  service:
    replicas: 1
    autoscaling:
      maxReplicas: 10
      triggers:
        - source: prometheus
          metricName: MetricName
          query: sum(rate(metric{label="value"}[1m]))
          threshold: 0.5
        - source: cpu
          type: Utilization
          value: 90

```

PandaDoc



Grafana & Prometheus

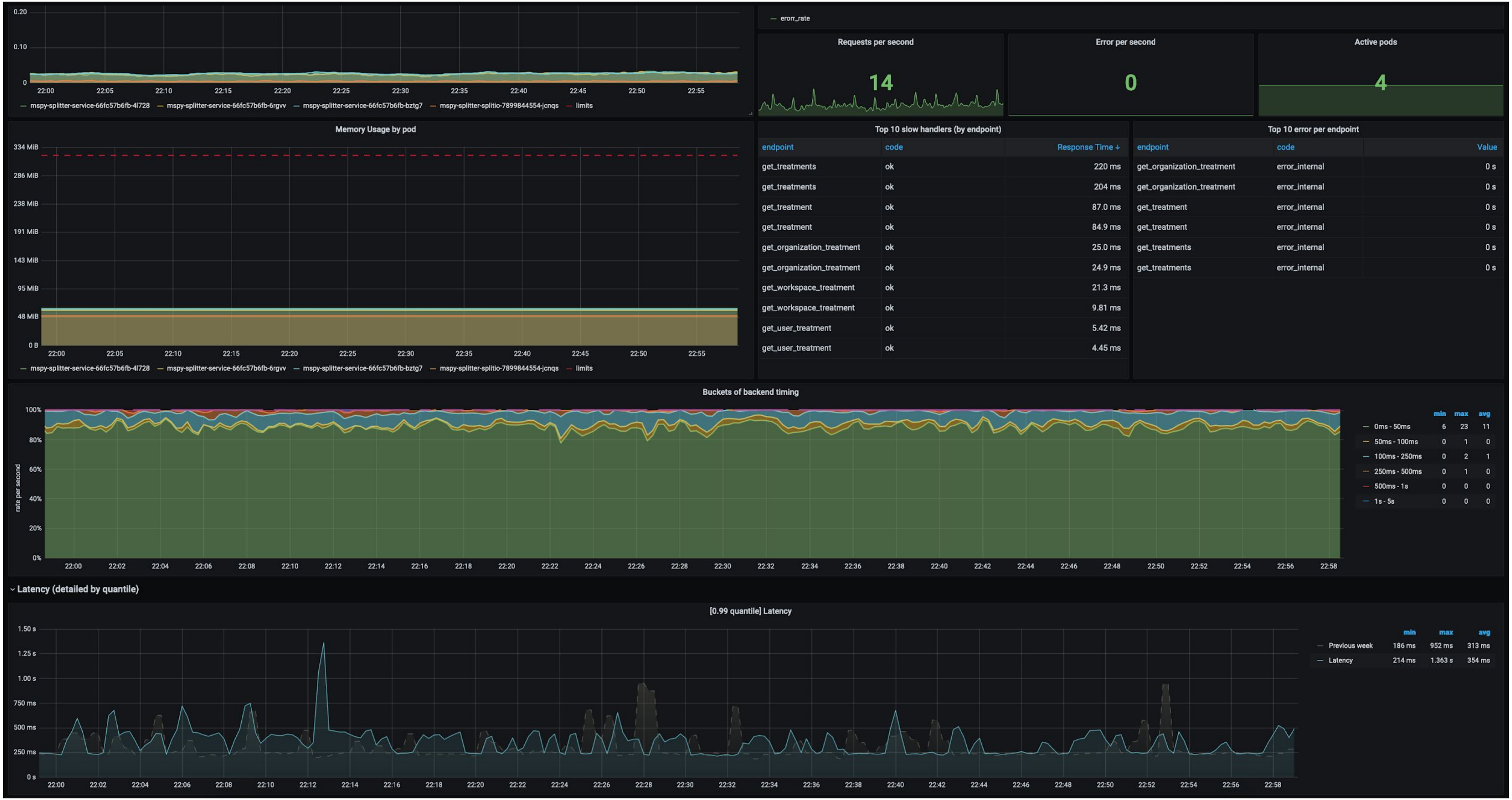
- Everything in one repository
- alerts/l2.yaml
- dashboards/queues.json
- Any alerts (as alert rules) or dashboards (as config maps) definitions can be deployed to Kubernetes and applied by **Prometheus operator**



```yaml
include:
  - project: platform/ci/k8s-pandakube
    ref: master
    file: pipelines/deploy-migrate.yml
  - project: platform/ci/k8s-pandakube
    ref: master
    file: pipelines/post-deploy-dashboards.yml
  - project: platform/ci/pipelines
    ref: master
    file: docker/uber.yml
  - project: platform/templates/template-pdms-service
    ref: master
    file: pipelines/uber.yml

stages:
  - prepare
  - quality
  - build
  - deploy-stagings
  - deploy-production
  - post-deploy

variables:
  CI_DEPLOY_GRAFANA_PROVISION_MS_APP_METRICS_DASHBOARD: "true"
  CI_PDMS_SERVICE_PYTHON_IMAGE_VERSION: "3.8"
  DOCKER_SERVICE_PIPELINE_IMAGE: $ARTIFACTORY_DOCKER_REGISTRY/$CI_PROJECT_PATH:$CI_COMMIT_REF_SLUG-$CI_PIPELINE_ID
```

Grafana & Prometheus

- Shared Grafana dashboard: CPU/memory usage; latency by percentiles; RPS; top slowest requests and more



Top 10 slow handlers (by endpoint)

endpoint	code	Response Time ↓
get_treatments	ok	220 ms
get_treatments	ok	204 ms
get_treatment	ok	87.0 ms
get_treatment	ok	84.9 ms
get_organization_treatment	ok	25.0 ms
get_organization_treatment	ok	24.9 ms
get_workspace_treatment	ok	21.3 ms
get_workspace_treatment	ok	9.81 ms
get_user_treatment	ok	5.42 ms
get_user_treatment	ok	4.45 ms

Top 10 error per endpoint

endpoint	code	Value
get_organization_treatment	error_internal	0 s
get_organization_treatment	error_internal	0 s
get_treatment	error_internal	0 s
get_treatment	error_internal	0 s
get_treatments	error_internal	0 s
get_treatments	error_internal	0 s

	min	max	avg
0ms - 50ms	6	23	11
50ms - 100ms	0	1	0
100ms - 250ms	0	2	1
250ms - 500ms	0	1	0
500ms - 1s	0	0	0
1s - 5s	0	0	0

	min	max	avg
Previous week	186 ms	952 ms	313 ms
Latency	214 ms	1.363 s	354 ms

Migration HOWTO

Apes together strong.

Migration HOWTO

- Dedicated team:
  - 1 DevOps
  - 1 Java
  - 2 Python
  - 2 QA
- Prepare shared Helm chart and Gitlab pipelines
- Prepare list of services to migrate
- **More than 50 services in 5 months**

Migration HOWTO

- Fast and furious
- Lack of knowledge sharing

(Epic)fails

(Epic)fails

- Pay attention to resources and number of replicas
- Better to overprovision and then tune based on live metrics

Takeaways

Takeaways

- 20%-30% faster deployments
- 5% more deployments ¯_(ツ)_/¯
- No more than 1 hour to start and deploy new service
- Better development experience
    - Improved CI/CD pipeline
    - Simple rollbacks
    - Easy scaling and resources management
    - Easy to deploy alerts and dashboards

Future plans

Future plans

- Dynamic environments 🔥

PandaDoc

Thank you

101 California St #3975,
San Francisco, CA 94111,
United States