Run Cloud Product in Kubernetes at

Developer's Laptop

Production

Client's Hardware

by Aleksandr Shinkarev

WYSIWYG

Today’s Roadmap

- Story of My Life (One Direction)
- What Do You Mean? (Justin Bieber)
- No Fear (Rasmus)

Kermit The Frog Map GIF by Muppet Wiki

Our Use Case

New Kubernetes Cloud App

As an Architect

Need to build a new app

So that at the end we can host it anywhere

Why do you need to host anywhere?

No decision on hosting

→ need to be ready for everything

→ Cloud or Intranet

What ~~if~~ else?

- Client needs to play with the app
- Teams needs to start somehow

First thing that comes to mind

Let’s spin up a Testing environment?

- Nowhere to spin up by our Client
- Don’t want to create Dev/Test/Prod environment by ourselves

Confidentiality risks for Client’s data are too high

Bottom Line

Well let 's run
the whole project
locally in Kubernetes.

Kubernetes locally?

BTW what about Docker Compose?

- Hard to support both Kubernetes and Docker Compose setup
- Don't want to setup network twice
- Want homogeneous development and deployment experience

Beginners Pause

Docker vs Kubernetes

The Illustrated Children's Guide to Kubernetes

Ok, what can we use for ~~local~~ k8s?

Minikube

k3d

k3s

Docker Desktop

and whatever...

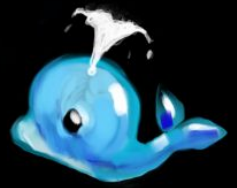
Is it the kindest?

We picked Kind because...

Objective Reasons

1. Loved its documentation at first glance
2. Was super easy to follow its getting started
3. Well, our client mentioned this one

+ k8s cluster in Docker



Kind

Deploy me to k8s

Helm

- packaging for Kubernetes
- deploy a **single** service to Kubernetes

Helm

Deploy them all but locally...

Helmfile + helm-diff

- templating over helm templates
- deploy not a single service but multiple

```
 containers:
   - name: nginx
-    image: docker.io/pandyz/to-dos-ui:latest
-    imagePullPolicy: "Always"
+    image: local-to-dos-ui:0.0.1
+    imagePullPolicy: "Never"
     env:
       - name: BITNAMI_DEBUG
         value: "true"
       - name: NGINX_HTTP_PORT_NUMBER
         value: "80"
     envFrom:
       - configMapRef:
           name: to-dos-ui-nginx
```

Helmfile

A long time ago in kubectl far away from simplicity

Lens

- Nice and easy UI to get started with
- Free for Personal use again...

```
PC – -bash
PC > kubectl get pods
```

Lens

To Docker or not to Docker?

https://www.imdb.com/title/tt0024660/mediaviewer/rm3321999360/

1 Cross-Platform

2 Dockerized

So many tool names...

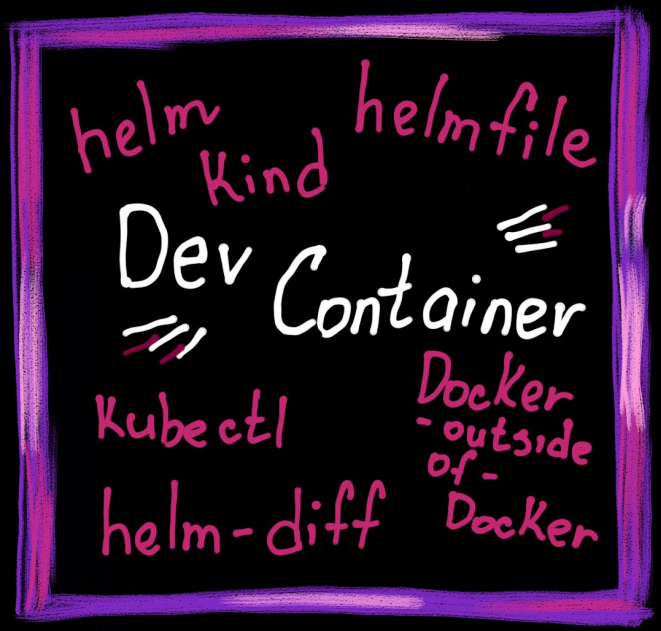
What do we use then?

Three Tools: helm, helmfile, helm-diff for the Elven-kings under the sky,

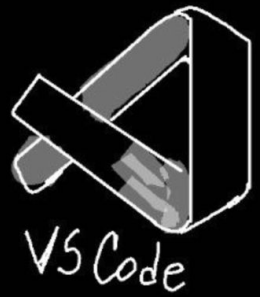
One Kind to rule them all. One Lens to find them,

One Dev Container to bring them all and in the darkness bind them

Dev Containers

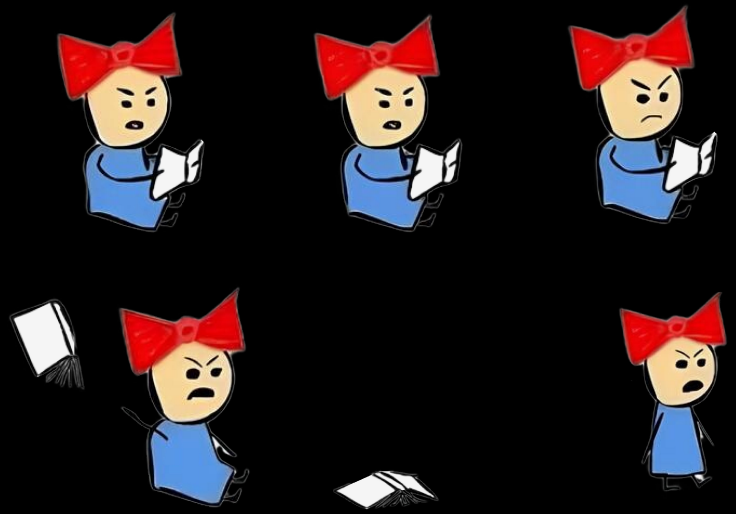


from



Our colleagues speech about Dev Containers (2023)

Let me write Dockerfile!



https://pikabu.ru/story/tyomnaya_storona_javascript_4477613

Once and for all

Features!

•••

```
"features": {
        "ghcr.io/devcontainers/features/docker-outside-of-docker:1.4.5": {
                "version": "20.10",
                "enableNonRootDocker": "true",
                "moby": "true"
        },
        "ghcr.io/devcontainers/features/kubectl-helm-minikube:1.1.9": {
                "version": "1.31.2",
                "helm": "3.16.2",
                "minikube": "none"
        },
        "ghcr.io/devcontainers-contrib/features/kind:1.0.14": {
                "version": "0.25.0"
        },
        "ghcr.io/nucleuscloud/devcontainer-features/helmfile:0.1.0": {
                "version": "v0.169.1"
        }
 }
```

•••

.devcontainer/devcontainer.json

Give me some juice!

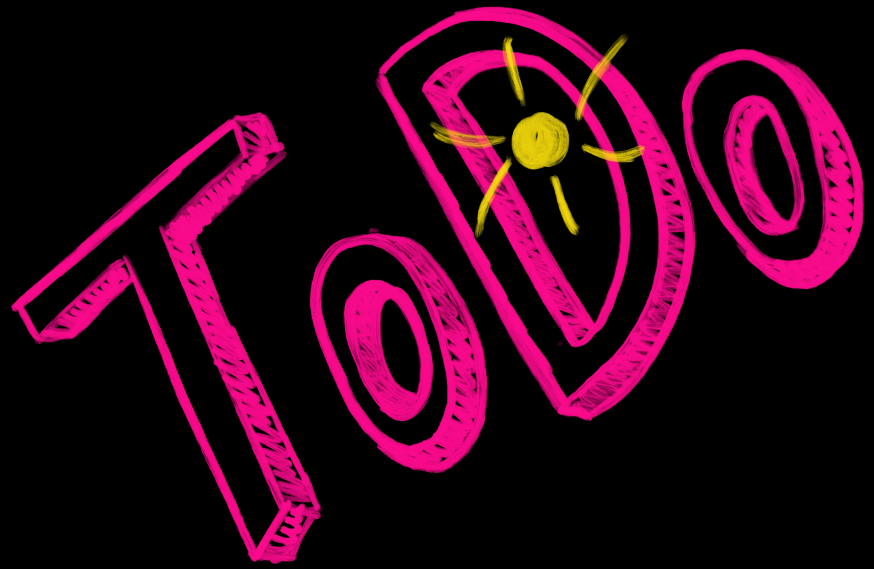
PNG image from web-site ru.pngtree.com

First Example

Your Favorite Web App

UI

API



and that is it.

Local Env

No https

Manual hosts change

```
127.0.0.1     to-dos.local.tourmalinecore.internal
```

- TO-DOS-LOCAL-ENV [DEV CONTAINER: LOCAL
  - .devcontainer
    - devcontainer.json
  - deploy
    - environments / local
      - values.yaml.gotmpl
    - helmfile.yaml
    - values-ingress-nginx.yaml.gotmpl
    - values-to-dos-ui.yaml.gotmpl
    - values.yaml.gotmpl
  - .gitattributes
  - .gitignore
  - .to-dos-cluster-kubeconfig
  - kind-local-config.yaml
  - LICENSE
  - README.md

to-dos-local-env

Kind

Create Cluster

kind-local-config.yaml

```yaml
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  extraPortMappings:
  - containerPort: 30080
    hostPort: 40080
    listenAddress: "0.0.0.0"
```

kind-local-config.yaml

Kind

Create Cluster

```
kind create cluster
  --name to-dos
  --config kind-local-config.yaml
  --kubeconfig ./.to-dos-cluster-kubeconfig
```

Repo local kubeconfig
(not polluting global)

Kind

Create Cluster

```
Creating cluster "to-dos" . . .
✅Ensuring node image (kindest/node:v1.31.2)
✅Preparing nodes
✅Writing configuration
✅Starting control-plane
✅Installing CNI
✅Installing StorageClass
Set kubectl context to "kind-to-dos"

You can now use your cluster with:
kubectl cluster-info --context kind-to-dos
--kubeconfig ./.to-dos-cluster-kubeconfig
```

Deploy it!

Local Env Helmfile

```yaml
releases:
  - name: ingress-nginx
    labels:
      app: ingress-nginx
    wait: true
    chart: ingress-nginx/ingress-nginx
    version: 4.10.1
    values:
      - values-ingress-nginx.yaml.gotmpl
```

helmfile.yaml

Deploy it!

Local Env Helmfile

Pull remote helm chart from Git!

```yaml
- name: to-dos-ui
  labels:
    app: to-dos-ui
  wait: true
  chart: bitnami/nginx
  version: 15.3.5
  needs:
    - ingress-nginx
    - to-dos-api
  values:
    - git::https://github.com/TourmalineCore/to-dos-ui.git@/ci/values.yaml?ref=master
    - values.yaml.gotmpl
    - values-to-dos-ui.yaml.gotmpl
```

helmfile.yaml

Service Hosts its Prod Helm Chart

ci/values.yaml

```yaml
ingress:
  enabled: true
  hostname: "to-dos.tourmalinecore.com"
  path: ""
  annotations:
      cert-manager.io/cluster-issuer: letsencrypt
      nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
  ingressClassName: "nginx"
  tls: true
```

ci/values.yaml

Service Builds its Docker Image

- latest and commit tags
- amd64, arm64 platforms

Helmfile

Deploy to Local Env

```
helmfile cache cleanup &&
helmfile
      --environment local
      --namespace local
      -f deploy/helmfile.yaml
      apply
```

Everytime pull remote helm chart from Git from scratch!

```
UPDATED RELEASES:
NAME            NAMESPACE   CHART                          VERSION   DURATION
ingress-nginx   local       ingress-nginx/ingress-nginx    4.10.1       1m44s
to-dos-api      local       bitnami/nginx                  15.3.5         21s
to-dos-ui       local       bitnami/nginx                  15.3.5         15s
```

Show Me the River

Lens

Pods

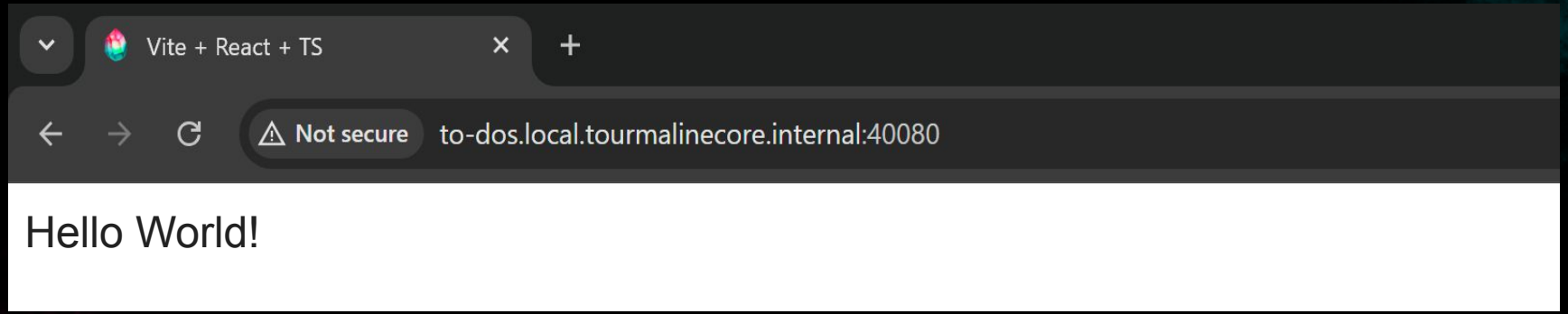
3 items

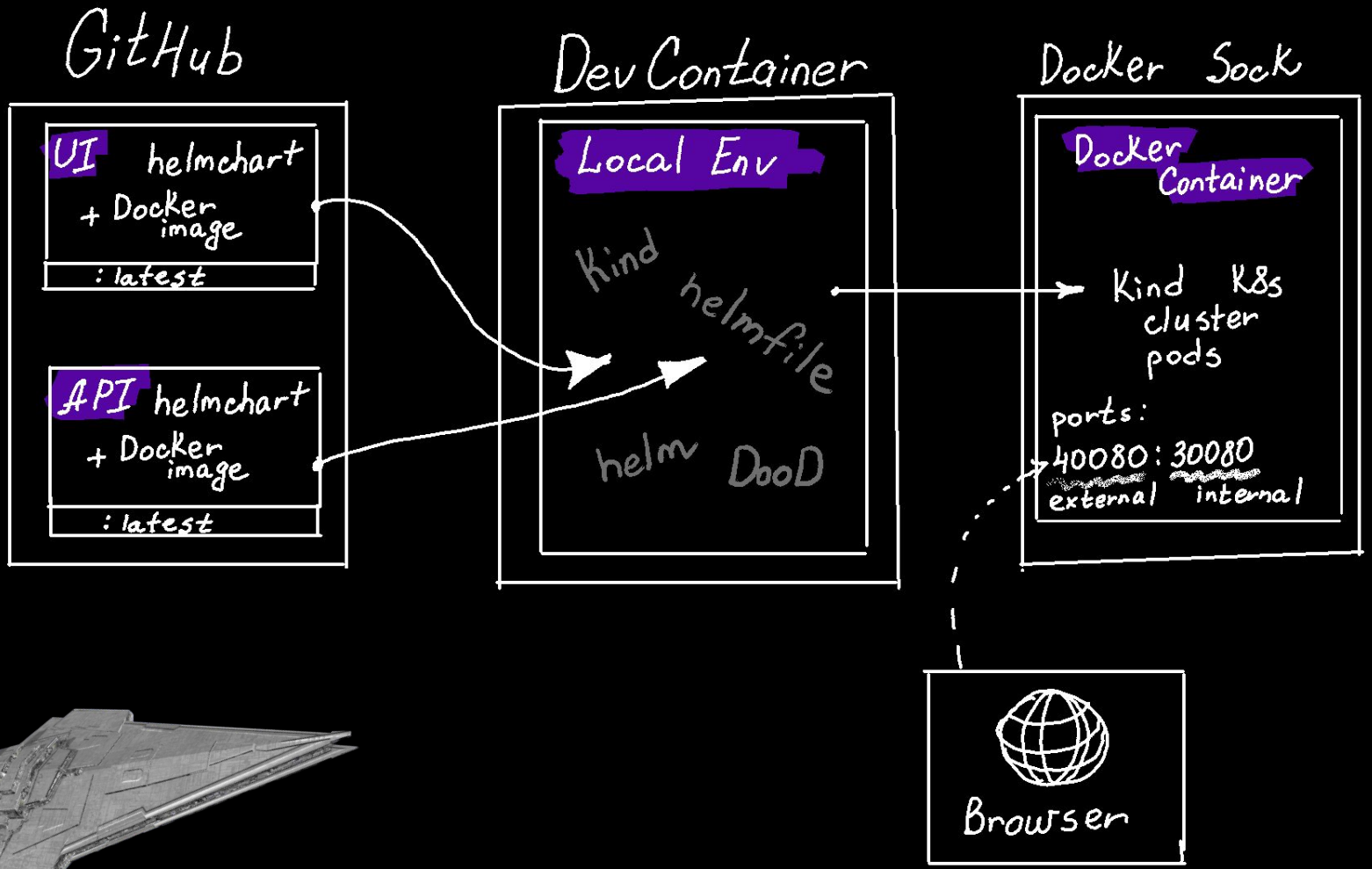
Namespace: local

Name	Namespace	Contai…	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status
ingress-nginx-controller-b7	local		N/A	N/A	0	ReplicaSet	to-dos-cor	Burstable	5m8s	Running
to-dos-api-nginx-85d9f8b78	local		N/A	N/A	0	ReplicaSet	to-dos-cor	Burstable	3m23s	Running
to-dos-ui-nginx-5ddbb9785	local		N/A	N/A	0	ReplicaSet	to-dos-cor	Burstable	3m2s	Running

GUI

Browser


Vite + React + TS
Not secure
to-dos.local.tourmalinecore.internal:40080
Hello World!


GitHub
UI helmchart
+ Docker image
: latest
API helmchart
+ Docker image
: latest
Dev Container
Local Env
Kind
helmfile
helm
DooD
Docker Sock
Docker Container
Kind K8s cluster pods
ports:
40080 : 30080
external
internal
Browser

You need Docker
& VSCode
& Dev Containers

1 kind create

2 helmfile apply

Day to day scenarios

1. Run the whole stack
2. Debug a service from IDE — killer feature
3. One service using local Docker image
4. One service using FB Docker image

3 One service using local Docker image

```
docker build -t local-to-dos-ui:0.0.1 .
```

```
kind load docker-image local-to-dos-ui:0.0.1 --name to-dos
```

values-to-dos-ui.yaml.gotmpl

```yaml
image:
  registry: ""
  repository: "local-to-dos-ui"
  tag: "0.0.1"
  pullPolicy: "Never"
```

Give this image to kind cluster

Problems Along the Way

This is the way!

Learning curve is something else

- Developers have no idea
- They have to learn k8s → Learn it locally in a safe way

Mad The Internet GIF by MOODMAN

Several Projects

Multiple Local Envs

to-dos-control-plane 34226bece649	kindest/node:v1.31.2	Running	40080:30080 40491:6443 Show less
real-world-control-plane 3c4013672364	kindest/node:v1.31.2	Running	40090:30080 30100:30100 44609:6443 Show less

We need more gold!

RAM is Key

- 8 GB - designer's macOS laptop to start the lightest product
- 16 GB - frontend developer's macOS laptop to start mid size with no Kafka or Elastic or Other beasts
- 32 GB - backend developer's Windows to spin up heavy Kafka & Elastic

Latest is sort of better...

Latest vs Fixed Commit

- Easy to forget to kill a pod when latest changed
- Easy(lazy) to forget to commit a new SHA (can be automated)

NOTE: The Kubernetes default pull policy is `IfNotPresent` unless the image tag is `:latest` or omitted (and implicitly `:latest`) in which case the default policy is `Always`. `IfNotPresent` causes the Kubelet to skip pulling an image if it already exists. If you want those images loaded into node to work as expected, please:

- don't use a `:latest` tag

and / or:

- specify `imagePullPolicy: IfNotPresent` or `imagePullPolicy: Never` on your container(s).

See Kubernetes imagePullPolicy for more information.

Quick Start

they wanted the best...

Lazy Failing Local Env

- Growing Local Env and secrets creeping
- Not dockerized setup of the tools (only Windows)
- Docker Compose and k3d (kind alternative) in the same repo
- Trying to start Kafka and ElasticSearch (ours)

Again if process sucks people
“forget” to make changes there.

What if there are too many services?

- Evolve to Production with feature deployments
- Allow to turn off certain features like: https, search, telemetry, or even app's features

Heavy Services

- Docker Compose them? (Argh)

Renzo Before The Gym GIF

Local Env Hybrid Mode

We used cloud Kafka and Elasticsearch from Local Env.

You can use even Production resources!

Hidden Bonus is E2E Tests

Where are Prod and Client?

Prod Env

https

ingress

Creates resources that
are shared among services.

- TO-DOS-PROD-ENV [DEV CONTAINER: PROD
  - .devcontainer
    - devcontainer.json
  - .github / workflows
    - deploy.yml
  - deploy
    - cert-manager-cluster-issuer.yaml
    - helmfile.yaml
    - values-cert-manager.yaml.gotmpl
    - values-ingress-nginx.yaml.gotmpl
  - .gitattributes
  - .gitignore
  - LICENSE
  - README.md

to-dos-prod-env

Each Service Deploys Itself to Production

ci/helmfile.yaml

```yaml
releases:
  - name: to-dos-ui
    labels:
      app: to-dos-ui
    wait: true
    chart: bitnami/nginx
    version: 15.3.5
    values:
      - values.yaml
```

ci/helmfile.yaml

Where can I deploy?

1. k8s in VM
2. k8s in cloud
3. Bare metal k8s

What about On-premises?

On-premises = Local Env

Local, Dev, Prod

- Local + Prod → Distributed Monolith

Oh Dev, UAT, Stage, Pre-Prod don't hurt me
Don't hurt me, no more

What is love?

Who is it useful for?

- Startups
- R&D
- MVP
- Hackathons
- Small & Middle size products
- Limited budget for multiple cloud environments
- SaaS & On-premises Product Delivery

When You're Gone

Who is it not for?

- Big products of 20-50-100 microservices
- Tricky external dependencies (mocking?)
- Very limited resources

V. Govorkov. No! 1954

In the End

Conclusion

- Can rapidly spin up a web-based product
- Examples allow you to use it almost as is
- Everything is dockerized => cross-platform but depends on VSCode
- Growth and scalability is baked in
- Isolation on a product level is possible.
- **You can deploy it anywhere!**

Aleksandr Shinkarev

12 years of experience

CEO at Tourmaline Core

to-dos-local-env

real-world-local-env

WIP

pelican-local-env

Follow us

Web site

tourmalinecore.com

Vk

tourmalinecore

Design by

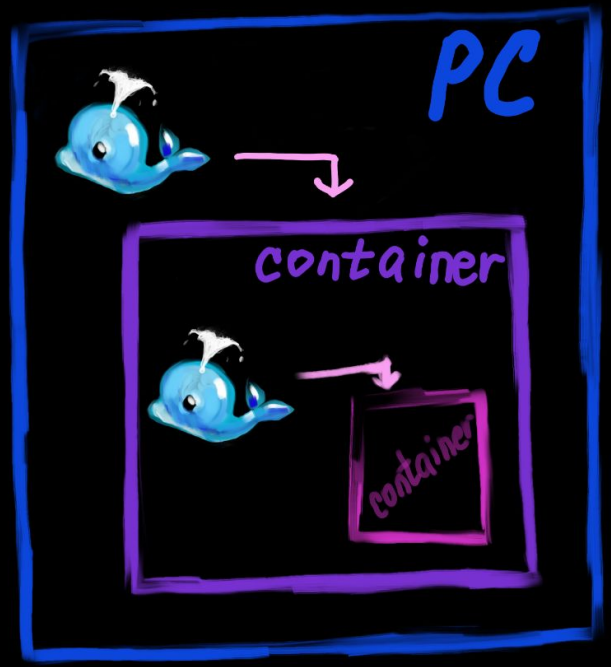
Anastasia Tupikina

Maria Yadryshnikova

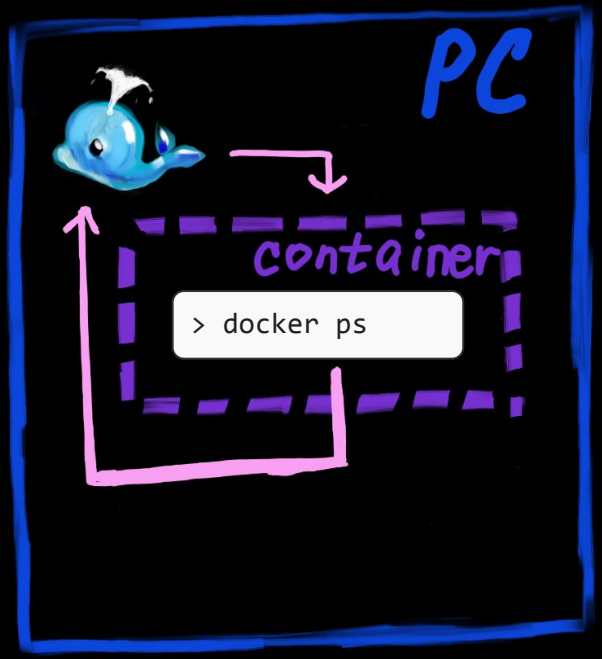
Give me more slides!

Dockerized, huh?

Docker in Docker

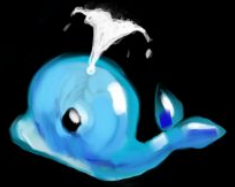


Docker out of Docker



DoOoOoOoD

Docker out of Docker



- You see them in Docker Desktop (observability)
- Partially re-use Docker images (re-usability)
- Eats less fast food (lighter)