OST
Eastern Switzerland
University of Applied Sciences

Challenge Task - CineShare

Module Distributed Systems & Blockchain

Simon Hager, Janis Wolf, Marius Zindel

17/05/2021

Department of Computer Science


vmware ESXi
docker Compose
kubernetes
Nodemailer
PostgreSQL
CERT MANAGER
RANCHER
HOSTPOINT
React
TS
Synology Virtual DSM License Pack
Ubuntu Server
Let's Encrypt
træfik
Synology
Hewlett Packard Enterprise
node
GitLab
UBIQUITI NETWORKS
Prisma
NGINX
IBM
METANET
docker
npm
Express JS

Agenda

- Demo
- Frontend
- Backend
- Deployment

Demo

- https://cineshare.ch

Challenge Task - CineShare

Frontend

Department of Computer Science

Simon Hager, Janis Wolf, Marius Zindel

Module Distributed Systems & Blockchain

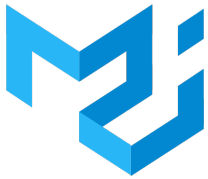
17/05/2021

OST
Eastern Switzerland
University of Applied Sciences

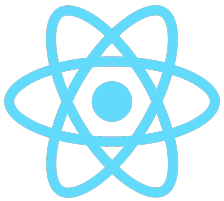
Frontend

Overview – Languages and Technologies

npm



Material-UI

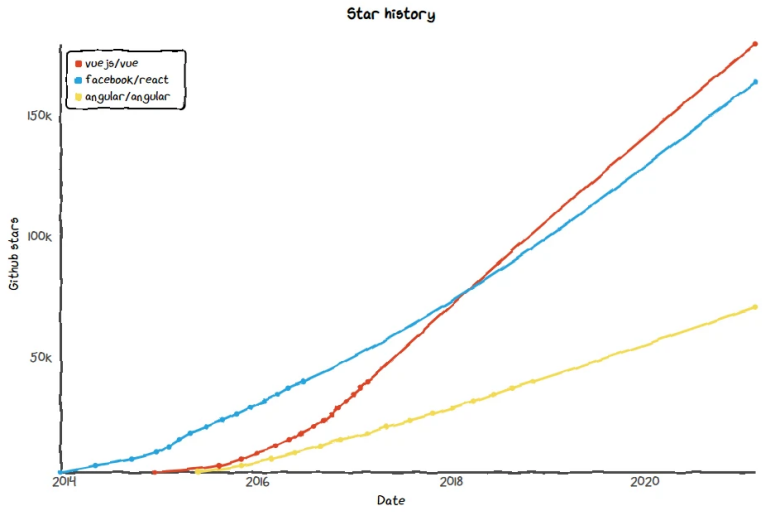


React.js

JSX

What is React.js?

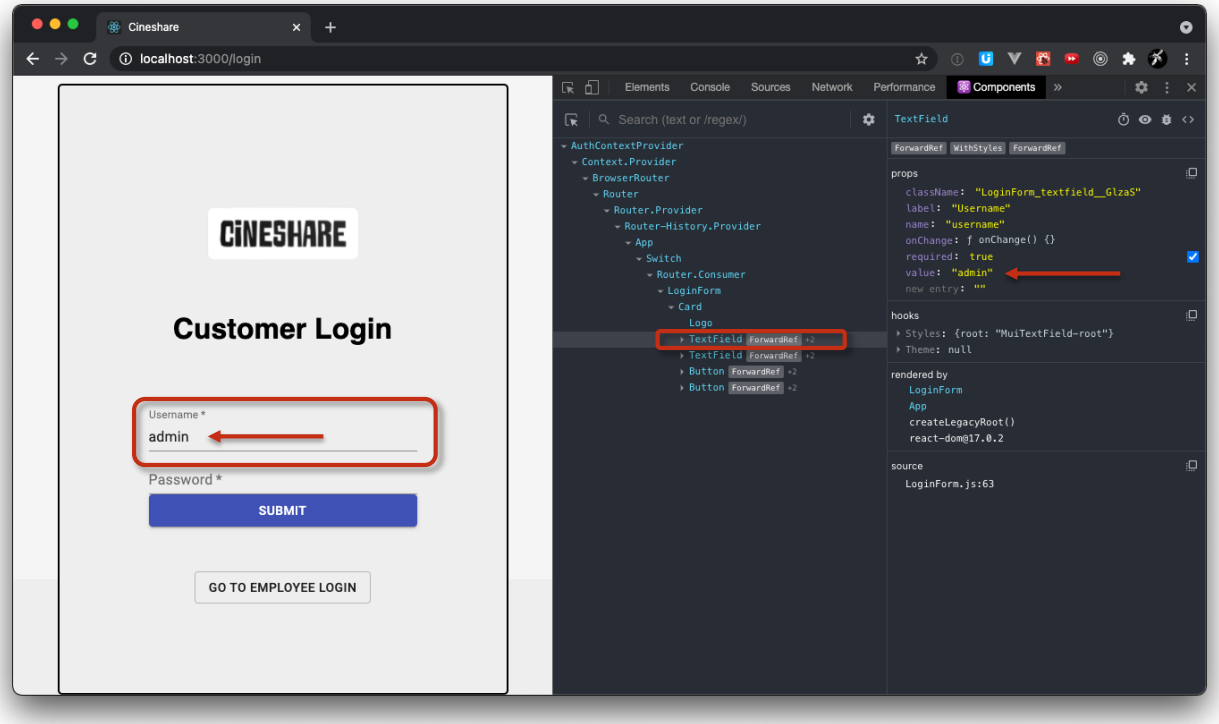
- “A JavaScript library for building user interfaces”
- Based on components -> High reusability
- SPA (Single Page Application) -> Site never refreshs
- Developed by Facebook
- Alternatives: Angular or Vue
- https://www.codeinwp.com/blog/angular-vs-vue-vs-react/



Number of stars on GitHub projects for Angular, React, and Vue

Frontend

React Example – Login



Frontend

Functionality – Employee

- Login, Logout
- Jobs CRUD
- Customer CRUD
- Employee CRUD

Frontend

Functionality – Customer

- Login, Logout
- Jobs (read only)

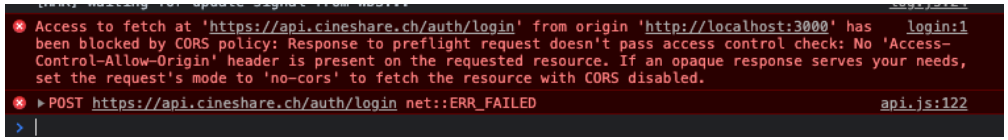
Functionality – Backlog

- Show statistics (Number of jobs, customers …)
- File upload for final / draft videos
- Automated creation of bills (and send / notify customer)
- Time tracking (how many hours spent on a job)
- …

Frontend

CORS

- “Cross-Origin Resource Sharing”
- “include credentials” -> sends cross-origin cookies
- https://de.wikipedia.org/wiki/Cross-Origin_Resource_Sharing




```
// Frontend
return fetch(`${api_fqdn_addr}${endpoint}`, {
  method: method,
  credentials: "include",
  headers: {
    Authorization: `Bearer ${token}`,
    "Content-Type": "application/json",
    Accept: "application/json",
  },
  body: JSON.stringify(params),
}).then(checkStatus);
```

```
// Backend
const corsOptions = {
  origin: [
    `${EnvVariableHelper.VARIABLES.FRONTEND_PROTOCOL}://${EnvVariableHelper.VARIABLES.FRONTEND_HOST}`,
    `${EnvVariableHelper.VARIABLES.FRONTEND_PROTOCOL}://www.${EnvVariableHelper.VARIABLES.FRONTEND_HOST}`,
  ],
  methods: ['GET', 'HEAD', 'PUT', 'PATCH', 'POST', 'DELETE'],
  credentials: true,
  maxAge: 600,
  allowedHeaders: ['Content-Type', 'Accept', 'Authorization', 'Origin'],
  optionsSuccessStatus: 204,
};
app.use(cors(corsOptions));
```

Challenge Task - CineShare

Backend

Department of Computer Science

Simon Hager, Janis Wolf, Marius Zindel

Module Distributed Systems & Blockchain

17/05/2021

OST
Eastern Switzerland
University of Applied Sciences

Backend

Overview – Languages and Technologies

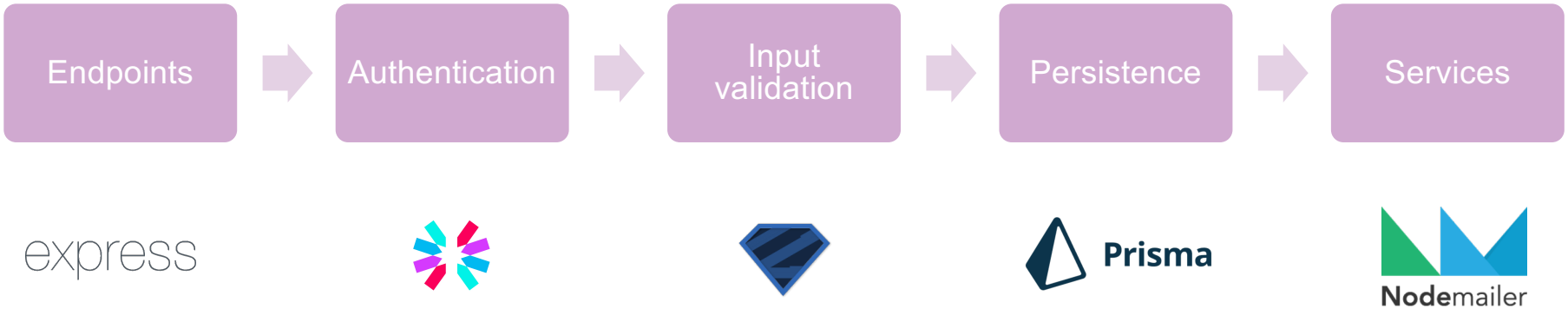
npm

TypeScript

Node.js

JavaScript

Overview – Backend stack



Overview – Dockerfile

Stage 1 - Builder

```
FROM node:15 AS builder
WORKDIR /usr/src/app

COPY package*.json ./
COPY tsconfig.json ./
RUN npm install

COPY src/ ./src/
COPY db/ ./db/

RUN npx prisma generate && npm run build
```

Stage 2 - Deployment image

```
FROM node:lts-alpine3.13
ARG sourcedir=/usr/src/app
COPY --from=builder $sourcedir/package*.json ./
COPY --from=builder $sourcedir/dist ./dist
COPY --from=builder $sourcedir/db ./db
RUN npm ci --only=production

COPY /resources ./resources

EXPOSE $BACKEND_PORT

CMD npm run update && npm start
```

express

Endpoints – Design decisions

- Same endpoints accessible for different roles
- Different responses based on the authenticated role/profile permissions

express

Endpoints

- /auth – Authentication
- /customers – CRUD Customer
- /employees – CRUD Employee
- /projects – CRUD Projects
- /stats – Read statistics

… more details

express

Endpoints – Authentication

/auth

- **POST** /auth/login
- **POST** /auth/refresh_token [1]
- **POST** /auth/logout [1]

1 Authenticated

OST

express

Endpoints – CRUD Customers/Employees/Projects

/customers [1][2]

- **GET** /customers
- **POST** /customers
- **GET** /customers/:id
- **PUT** /customers/:id
- **DELETE** /customers/:id

/employees [1][2]

- **GET** /employees
- **POST** /employees
- **GET** /employees/:id
- **PUT** /employees/:id
- **DELETE** /employees/:id

/projects [1][2]

- **GET** /projects
- **POST** /projects
- **GET** /projects/:id
- **PUT** /projects/:id
- **DELETE** /projects/:id

1 Authenticated
2 Authorized

express

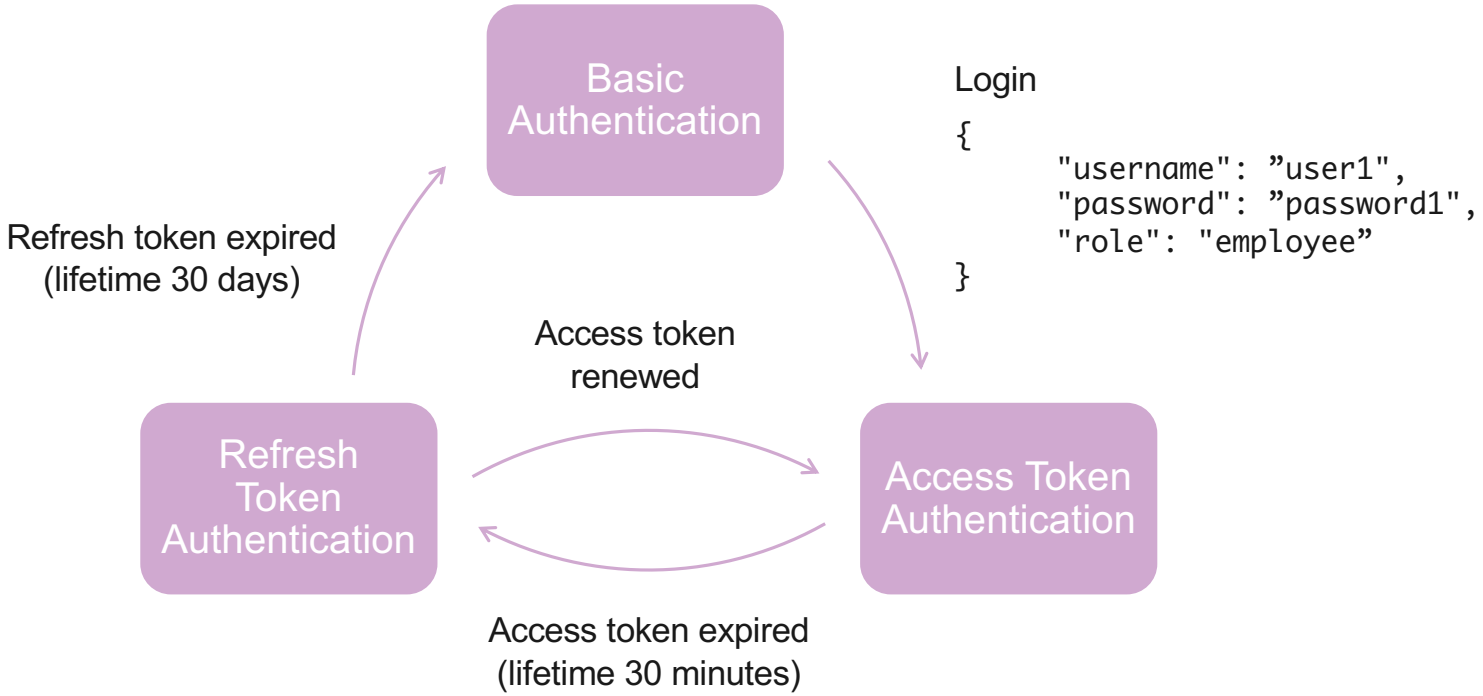
Endpoints – Read statistics

/stats [1][2]

- **GET** /stats/projectsThisYear
- **GET** /stats/earningsThisYear

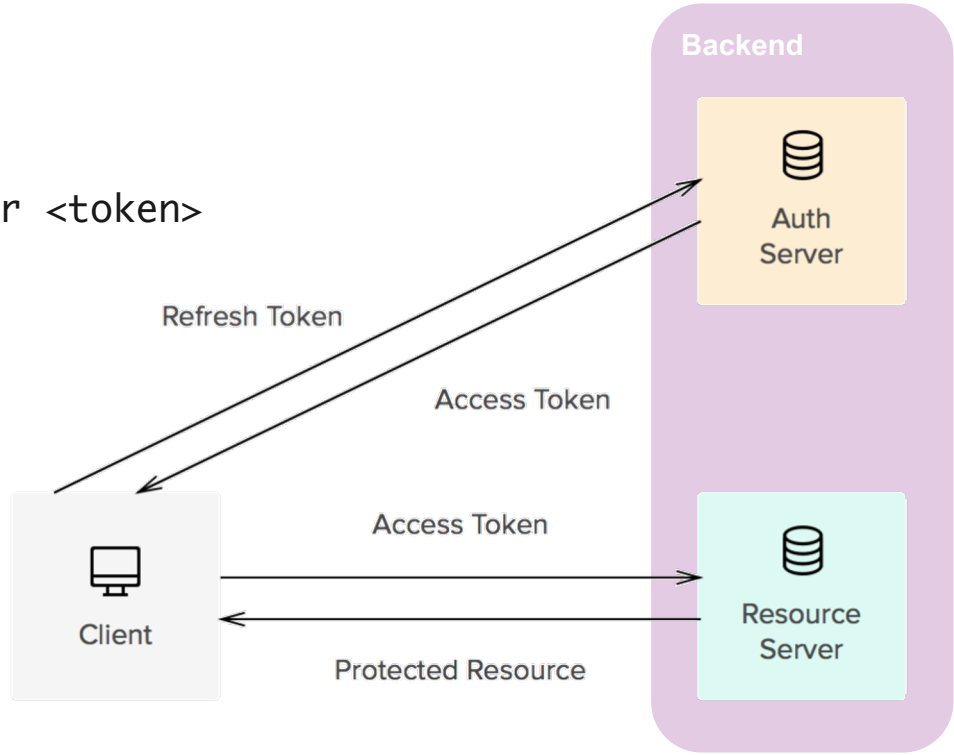
1 Authenticated
2 Authorized

Authentication – State diagram



Authentication – Token

- Stateless
- Json Web Token
- Client sends header `Authorization: Bearer <token>`



https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/

Backend

Input validation

- XSS Protection
- Store only high-quality data
- Prevent database errors

```
const Username = z
  .string()
  .transform((value) => value.toLowerCase())
  .refine(
     async (value) =>
       userListService
       .findUserByUsername(value)
       .then((user) => user === null),
       'Invalid input'
);


const Customer = z.object({
  firstname: z.string().nonempty(),
  lastname: z.string().nonempty(),
  street: z.string().nonempty(),
  city: z.string().nonempty(),
  zip: z.number().positive().min(3).max(10000),
  country: z.string().nonempty(),
  company: z.string().optional(),
  phone: z.string().optional(),
  mail: z.string().email().nonempty(),
});
```

src/declarations/validations.ts

- Status 400 Bad Request

```
{
     ”error": ”Invalid input",
}
```

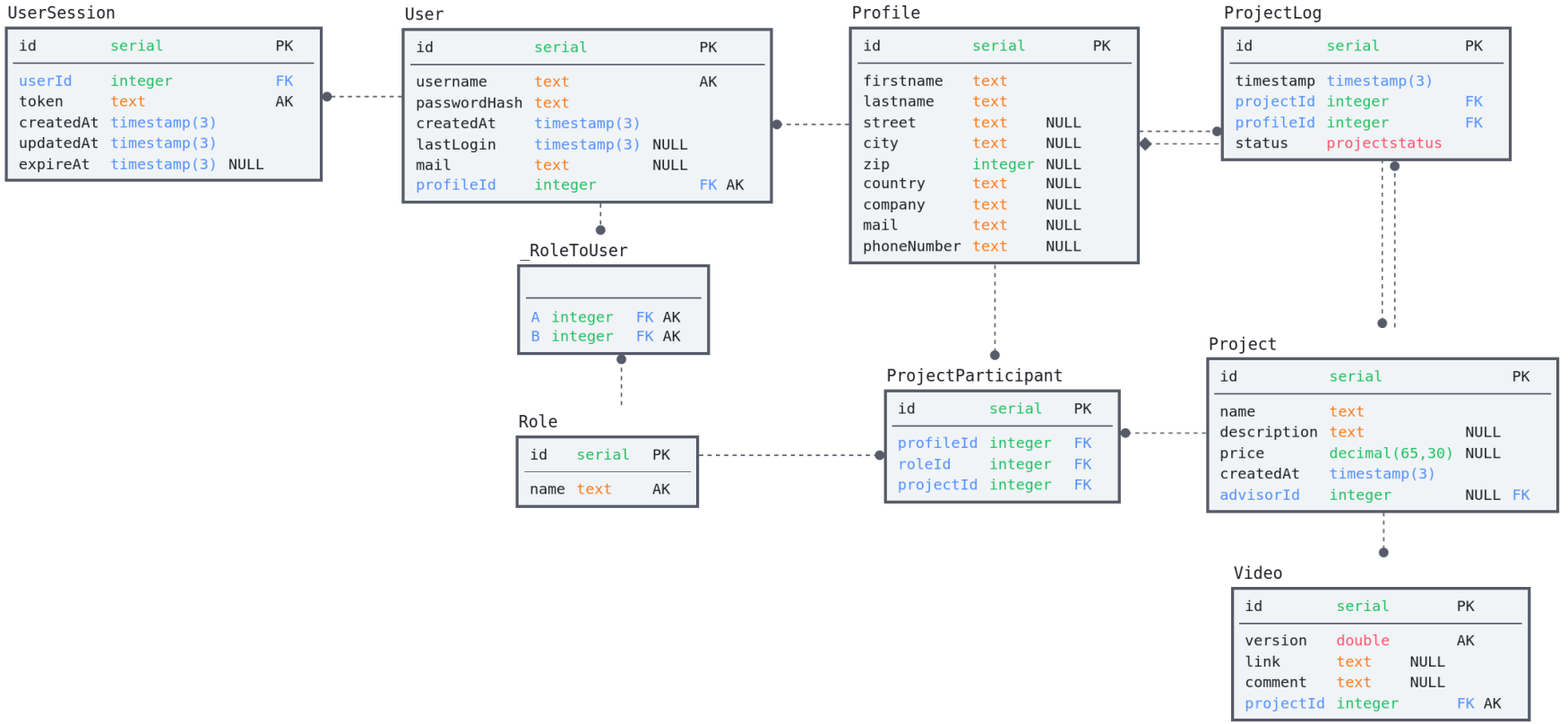
```
const result = await Customer.safeParseAsync({
  ...req.body,
});
```

src/controller/customerController.ts

Persistence – Design decisions

- Allows multiple open user sessions from different devices simultaneously
- Using same profile for customer and employees
- Project activities can be logged
- Standardized mapping to database (no sql commands required)

Persistence – Database schema



Persistence – Database schema

```
model User {
  id              Int             @default(autoincrement()) @id
  username        String          @unique
  passwordHash    String
  createdAt       DateTime        @default(now())
  lastLogin       DateTime?
  mail            String?
  activeSessions  UserSession[]
  roles           Role[]
  profile         Profile         @relation(fields: [profileId], references: [id])
  profileId       Int
}
```

db/schema.prisma

```
CREATE TABLE "User" (
    "id" SERIAL NOT NULL,
    "username" TEXT NOT NULL,
    "passwordHash" TEXT NOT NULL,
    "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "lastLogin" TIMESTAMP(3),
    "mail" TEXT,
    "profileId" INTEGER NOT NULL,

    PRIMARY KEY ("id")
);
```

db/migrations/…/migration.sql

Persistence – Database connection

```
datasource db {
  provider = "postgresql"
  url      = ”postgresql://username:password@hostname:5432/cineshare?schema=public"
}

generator client {
  provider = "prisma-client-js"
}
...
```

db/schema.prisma

```
private prisma = new PrismaClient();
```

src/services/db/customerList.ts

Persistence – Database queries

```
prisma.profile
   .create({
      data: {
         firstname,
         lastname,
         mail,
         user: {
            create: {
               username,
               passwordHash,
               mail,
               roles: {
                     connect: [{ id: 1 }],
                 },
              },
           },
      },
   })
   .catch((e) => {
      throw e;
   })
   .finally(async () => {
      await prisma.$disconnect();
});
```

src/services/db/customerList.ts

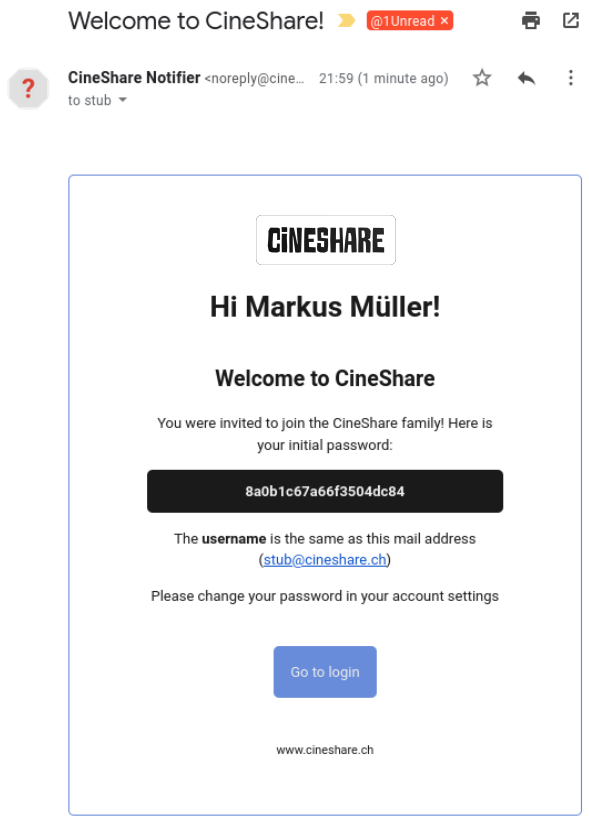
```
prisma.user
   .findFirst({
      where: {
         username: 'user1',
      },
      select: {
         id: true,
         username: true,
         lastLogin: true,
         roles: true,
         profileId: true,
      },
   })
   .catch((e) => {
      throw e;
   })
   .finally(async () => {
      await prisma.$disconnect();
});
```

src/services/db/userList.ts

Services – Mail delivery

Mail client
https://api.cineshare.ch

Mail server
ryan.metanet.ch



Demo

Login

- `curl -X POST -H "Content-Type: application/json" -d '{"username": "admin", "password": "admin", "role": "employee"}' https://api.cineshare.ch/auth/login`

Refresh token

- `curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer <refreshToken>" https://api.cineshare.ch/auth/refresh_token`

Access authenticated area

- `curl -X GET -H "Authorization: Bearer <accessToken>" https://api.cineshare.ch/customers/`

Challenge Task - CineShare

Deployment

Department of Computer Science

Simon Hager, Janis Wolf, Marius Zindel

Module Distributed Systems & Blockchain

17/05/2021

OST
Eastern Switzerland
University of Applied Sciences

Requirements

- Self hosted
- Kubernetes cluster
- Scalable Pods / Container
- Load balancing
- Automated certificate provisioning
- High availability
- Pull images from private GitLab repository
- Staged roll out of images
- Every housewife should be able to create a new deployment

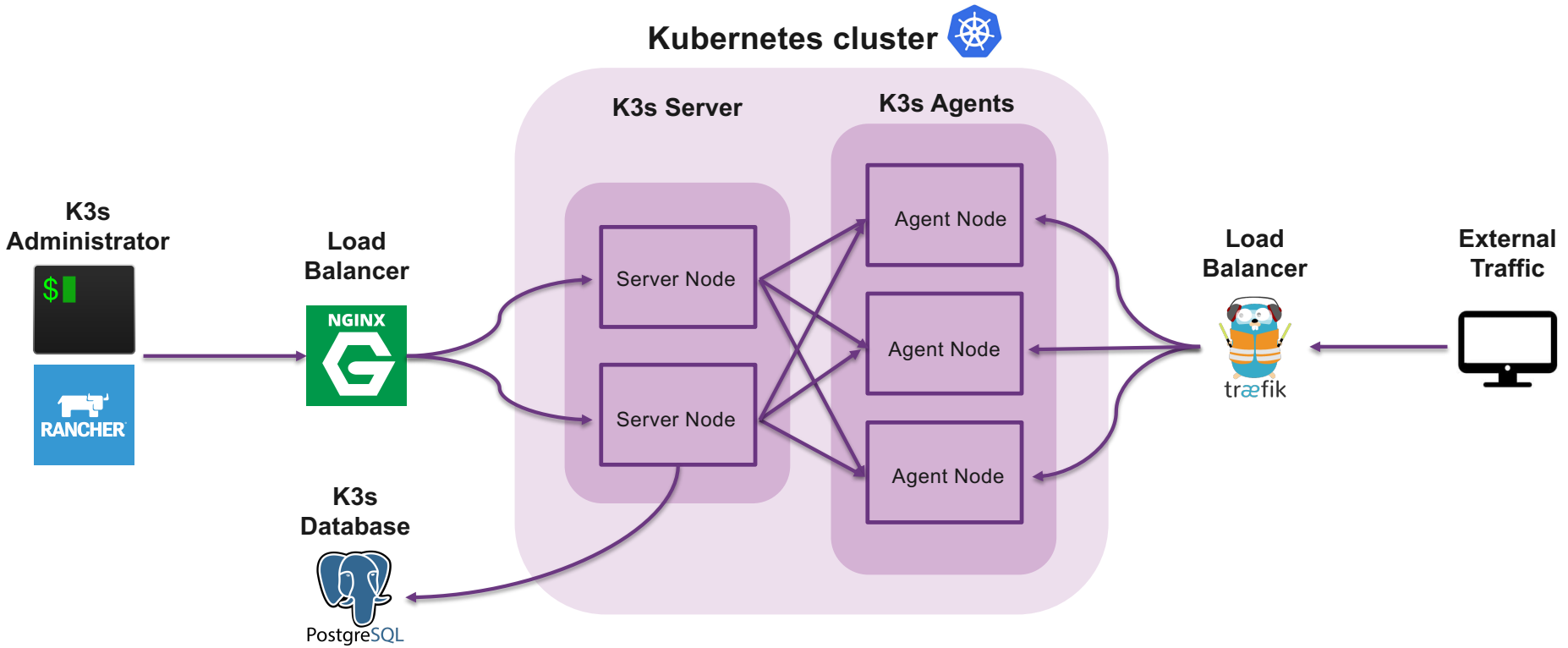
Deployment

Hosting

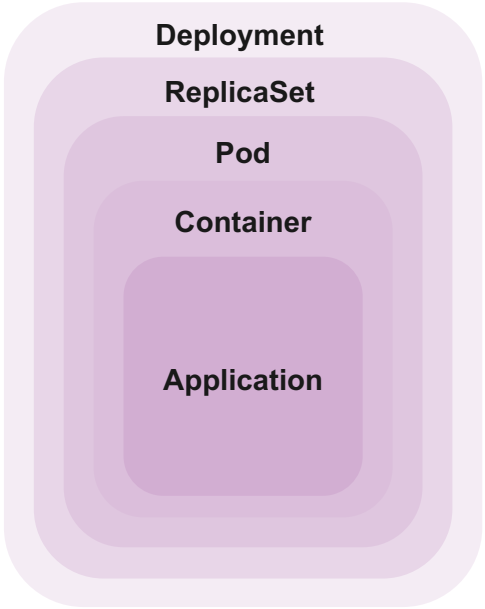
	OST-Datacenter	MFD-Datacenter	MFD-NAS
Hardware	unknown	Hewlett Packard Enterprise, IBM	Synology
Virtualization	vmware ESXi	vmware ESXi	Synology Virtual DSM License Pack
Operating System	Ubuntu Server	Ubuntu Server	Ubuntu Server
Container Orchestration	docker Compose, docker	kubernetes	kubernetes

Deployment

High Availability Architecture



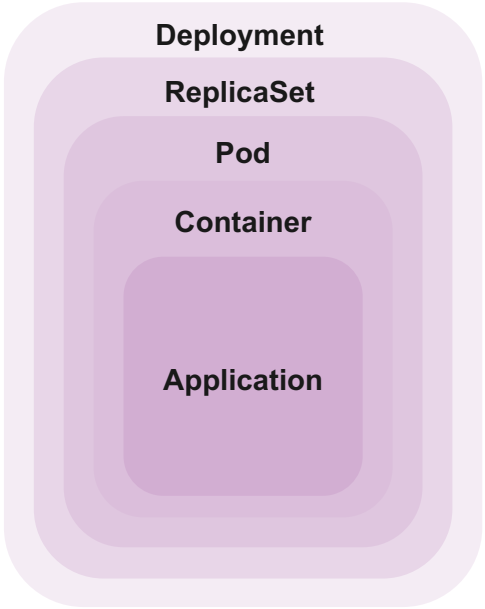
Cluster Architecture




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
  namespace: cineshare-namespace
  labels:
    app: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: registry.gitlab.ost.ch:45023/cineshare/frontend:latest
        imagePullPolicy: Always
        ports:
          - containerPort: 80
      imagePullSecrets:
      - name: regcred
```

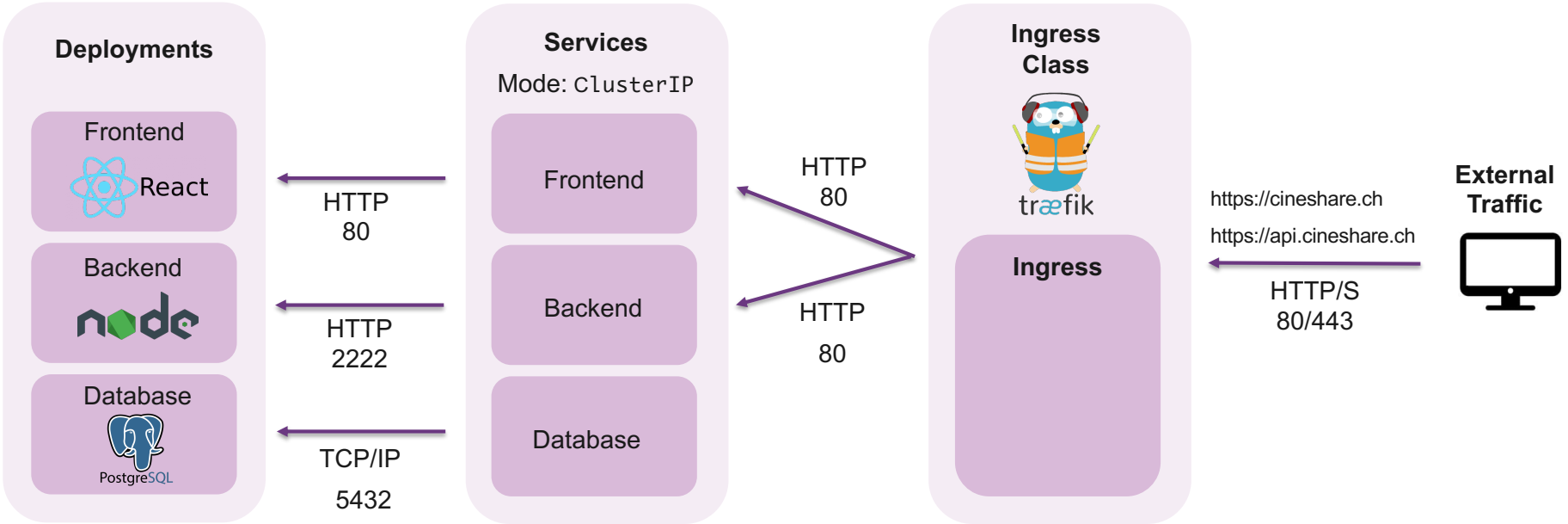
frontend-deployment.yaml

Scalable Pods

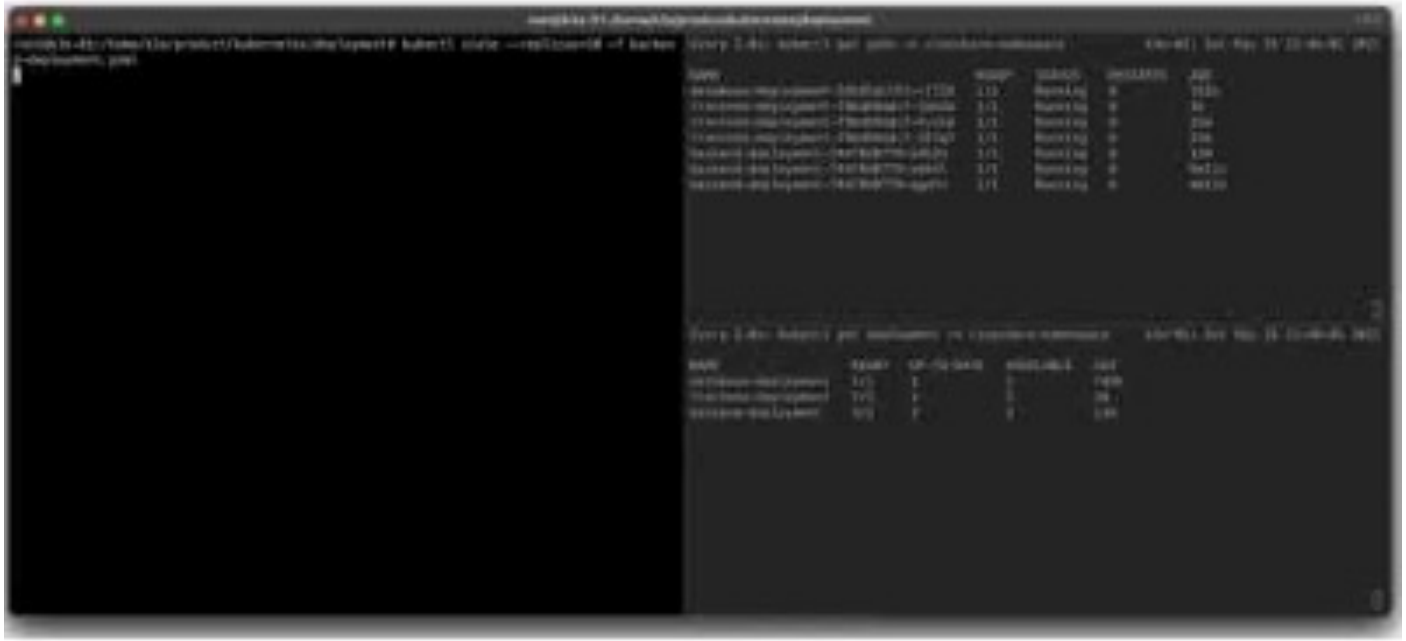


- Update the replicaset in your deployment.yaml
  - `vim foo.yaml`
  - Updates in file ….
  - Save file
  - `kubectl apply –f foo.yaml`
- Update directly via CLI
  - `kubectl scale --replicas=10 -f foo.yaml`

Load Balancing



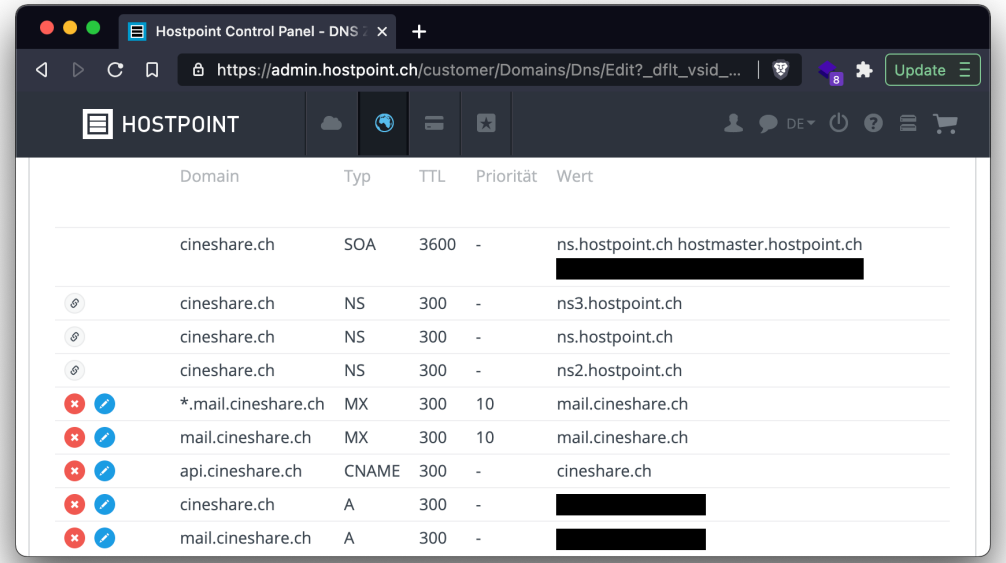
Load Balancing



Automated certificate provisioning

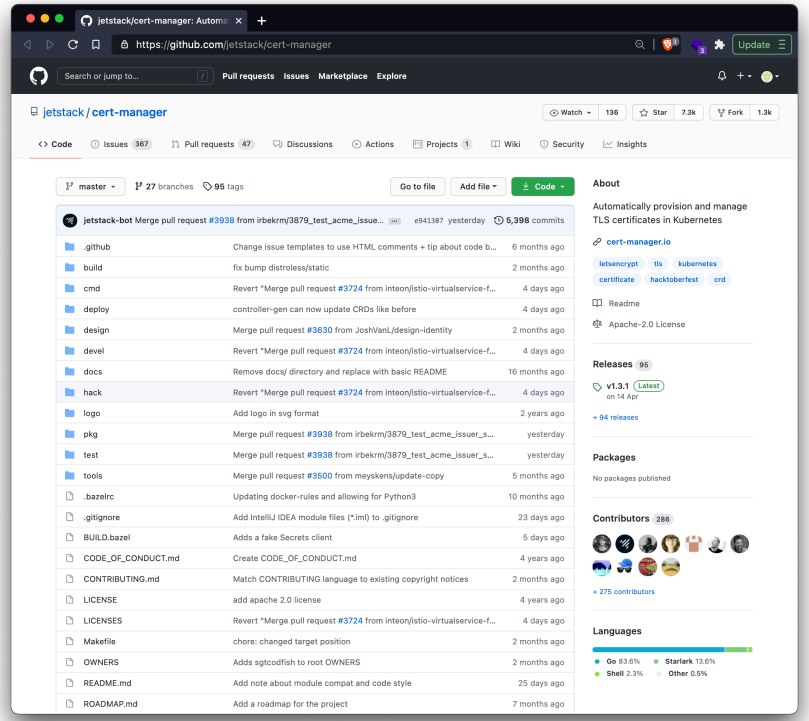
HOSTPOINT

- Domains
  - `cineshare.ch`
  - `api.cineshare.ch`
  - `mail.cineshare.ch`
- DNS Records hosted by Hostpoint
  - A Records
  - CNAME Record
  - MX Records



Automated certificate provisioning

- Open-Source Kubernetes Deployment



- Installation with Helm Chart
  - `helm repo add jetstack https://charts.jetstack.io`
  - `helm repo update`
- Installation with YAML Deployment
  - `kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.3.1/cert-manager.yaml`

- https://cert-manager.io/
- https://github.com/jetstack/cert-manager

Automated certificate provisioning

- Notice: Do not use the main Lets' Encrypt API for testing purpose!
- Use instead the staging environment
  - `https://acme-staging-v02.api.letsencrypt.org/directory`
- Otherwise, your IP will be blocked!

```yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: letsencrypt
  namespace: cineshare-namespace
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      name: letsencrypt
    solvers:
      - http01:
          ingress:
            class: traefik
```

issuer.yaml

```yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: certificate-main
  namespace: cineshare-namespace
spec:
  secretName: certificate-main
  issuerRef:
    kind: Issuer
    name: letsencrypt
  commonName: cineshare.ch
  dnsNames:
    - cineshare.ch
```

certificate-main.yaml

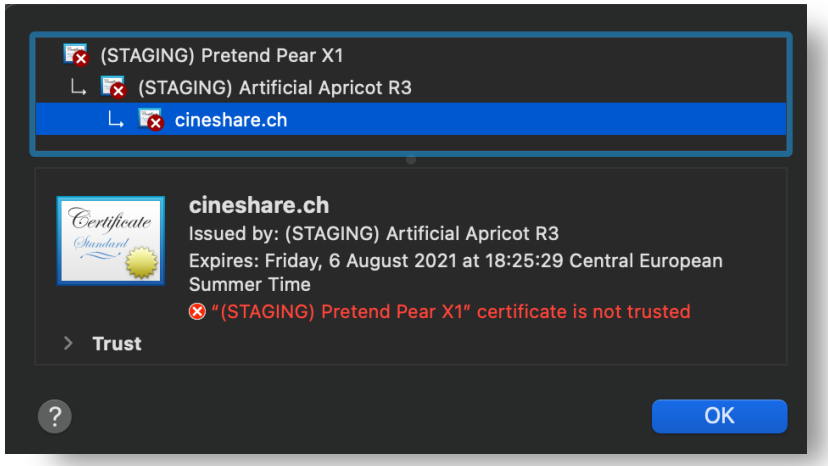
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend-ingress
  namespace: cineshare-namespace
  annotations:
    kubernetes.io/ingress.class: "traefik"
    traefik.ingress.kubernetes.io/frontend-entry-points: http, https
    traefik.ingress.kubernetes.io/redirect-entry-point: https
    cert-manager.io/issuer: "letsencrypt"
spec:
  tls:
    - hosts:
        - cineshare.ch
      secretName: certificate-main
```

frontend-ingress.yaml

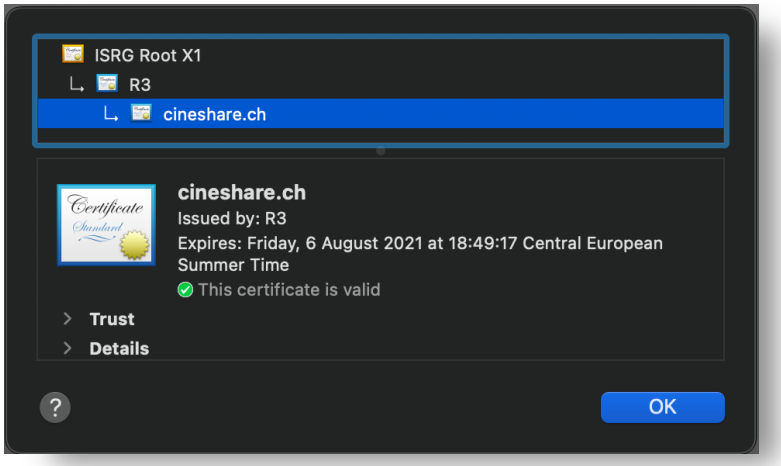
Deployment

Automated certificate provisioning

Staging Certificate

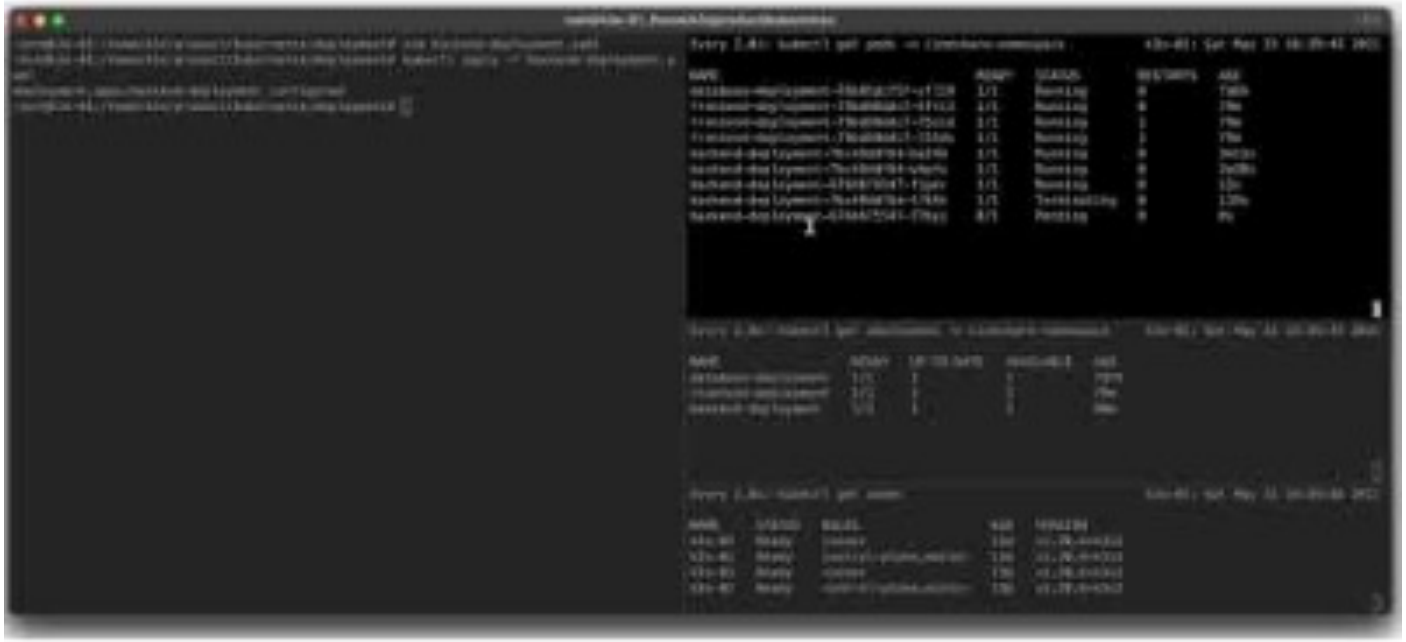


Valid Certificate



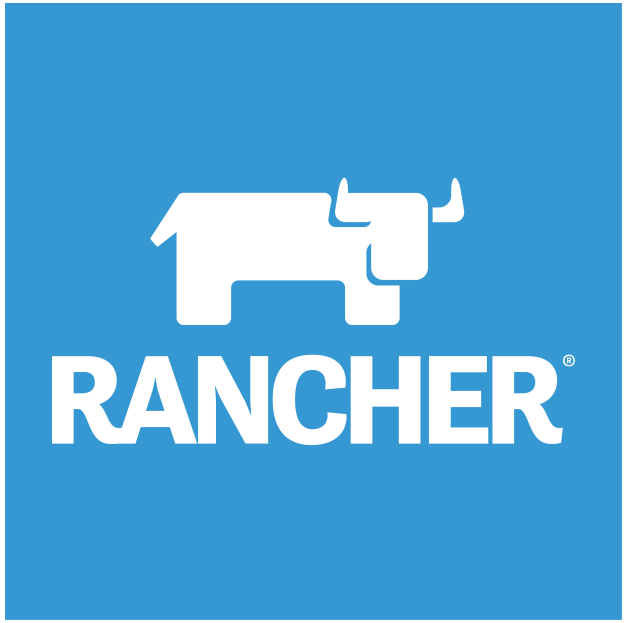
High Availability Test

Staged roll out of images



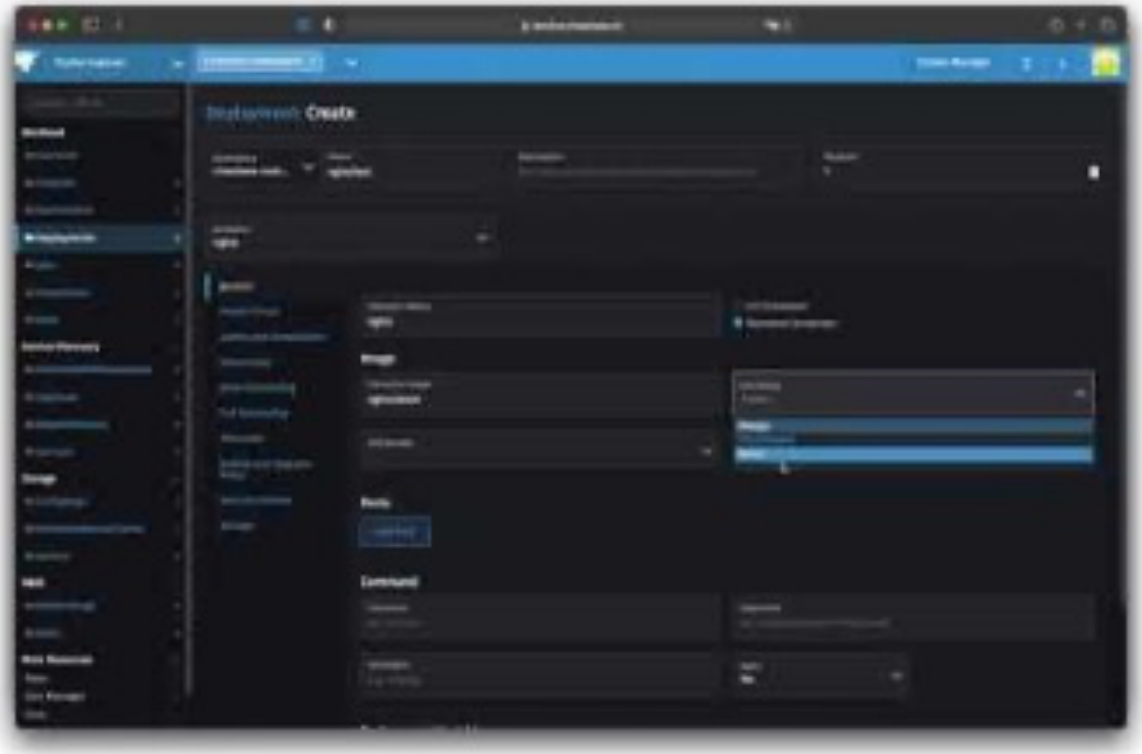
Every housewife should be able to create a new deployment

Every housewife should be able to create a new deployment



Deployment

Every housewife should be able to create a new deployment



OST
Eastern Switzerland
University of Applied Sciences

Challenge Task - CineShare

Module Distributed Systems & Blockchain

Simon Hager, Janis Wolf, Marius Zindel

17/05/2021

Department of Computer Science