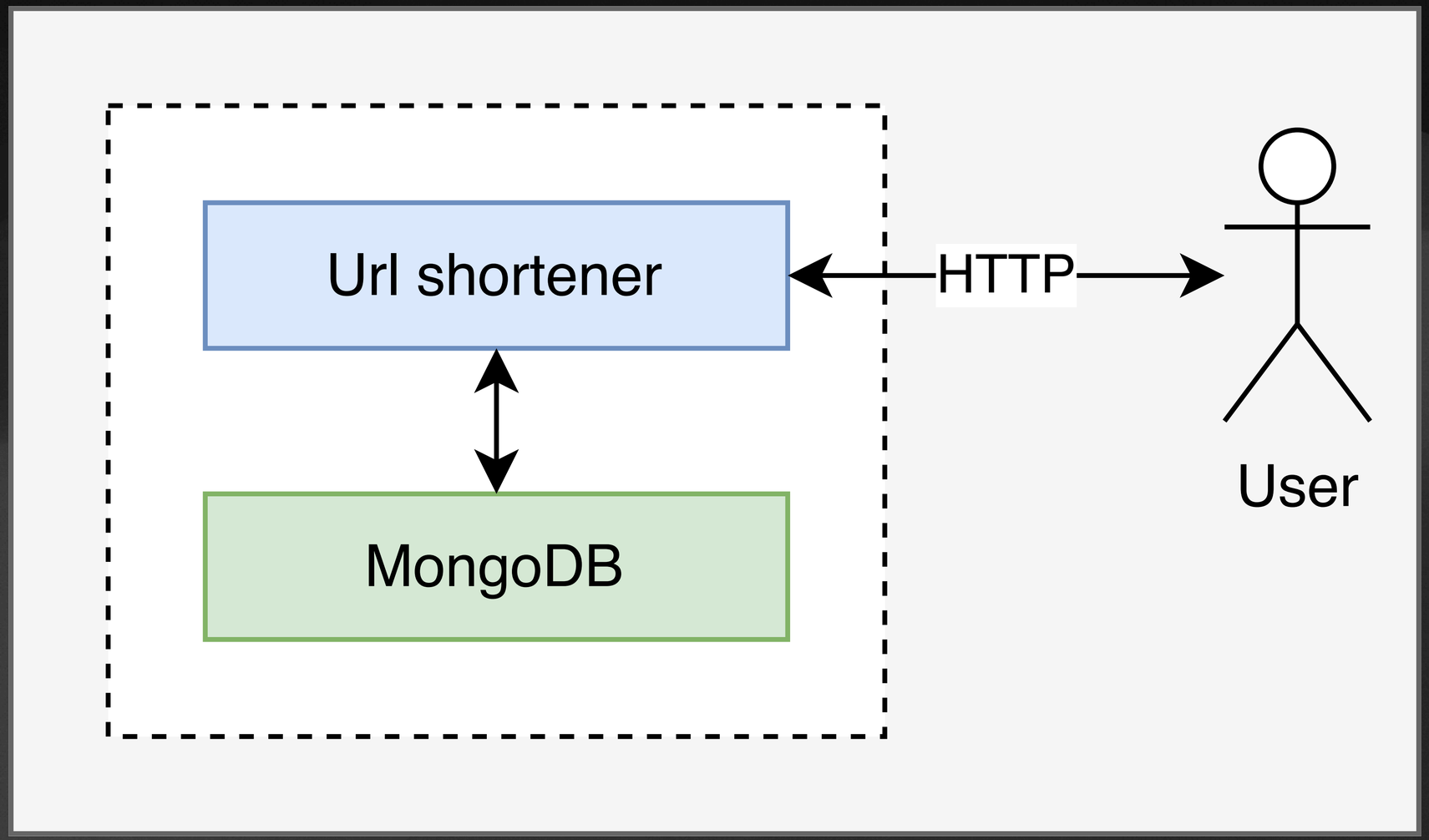
URL Shortener

Current architecture, design decisions, project demo and future tasks and architecture of an URL shortener application.

Carlos Pozuelo. October 2024

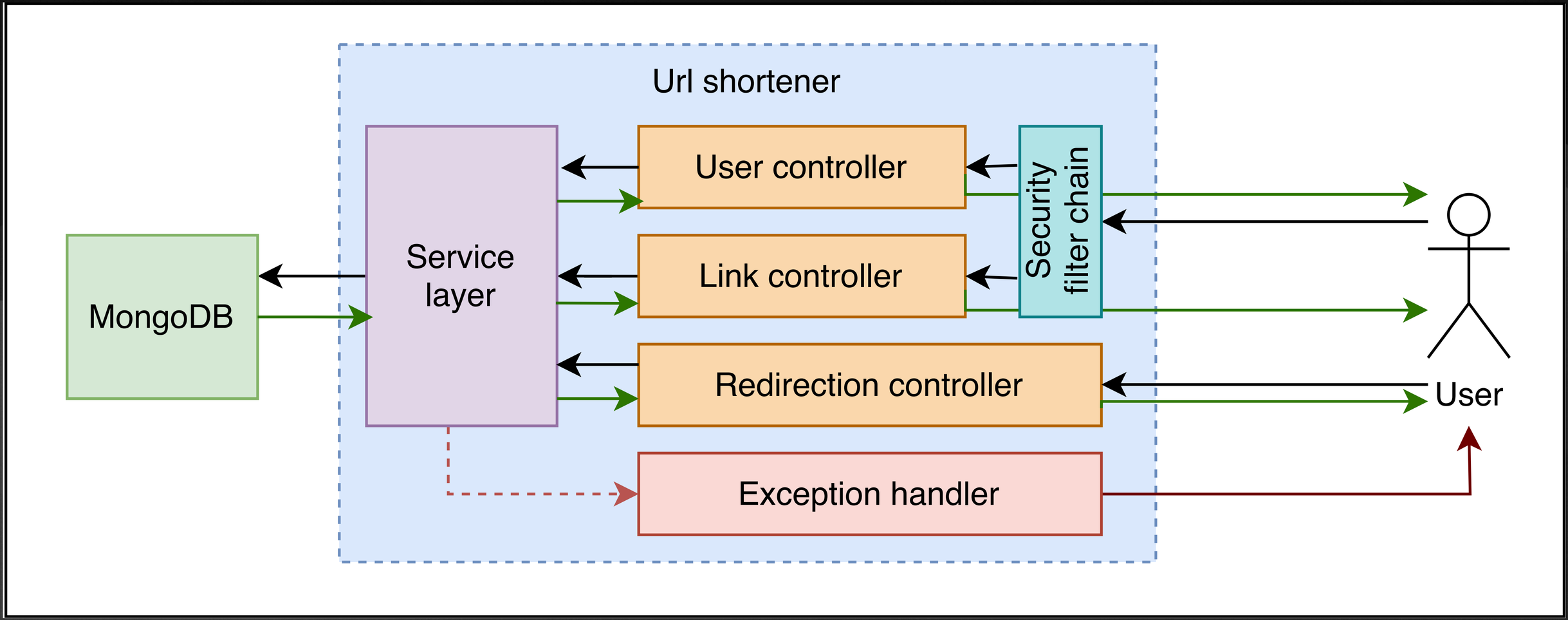
Current architecture

Current architecture



Current architecture

The URL shortener app



Current architecture

The MongoDB documents

MongoDB documents

User

- **id**
- password
- **username**
- roles

Note:
fields in blue are IDs
fields in orange are indexed.

Link

- **linkAlias**
- link
- isActive
- **user**

LinkStatistics

- visits

Users

```
{
  "id": ObjectId('.....1'),
  "username": "carlos",
  "password": "encoded password",
  "roles": [
    "USER"
  ]
}
```

```
{
  "id": ObjectId('.....2'),
  "username": "pozuelo",
  "password": "encoded password",
  "roles": [
    "USER"
  ]
}
```

Links

```
{
  "id": "yt",
  "link": "https://youtube.com",
  "isActive": true,
  "user": "carlos",
  "statistics": {
    "visits": 12
  }
}
```

```
{
  "id": "gl",
  "link": "https://google.com",
  "isActive": true,
  "user": "pozuelo",
  "statistics": {
    "visits": 101982
  }
}
```

Demo & code

Future architecture

Tasks to be done after the MVP

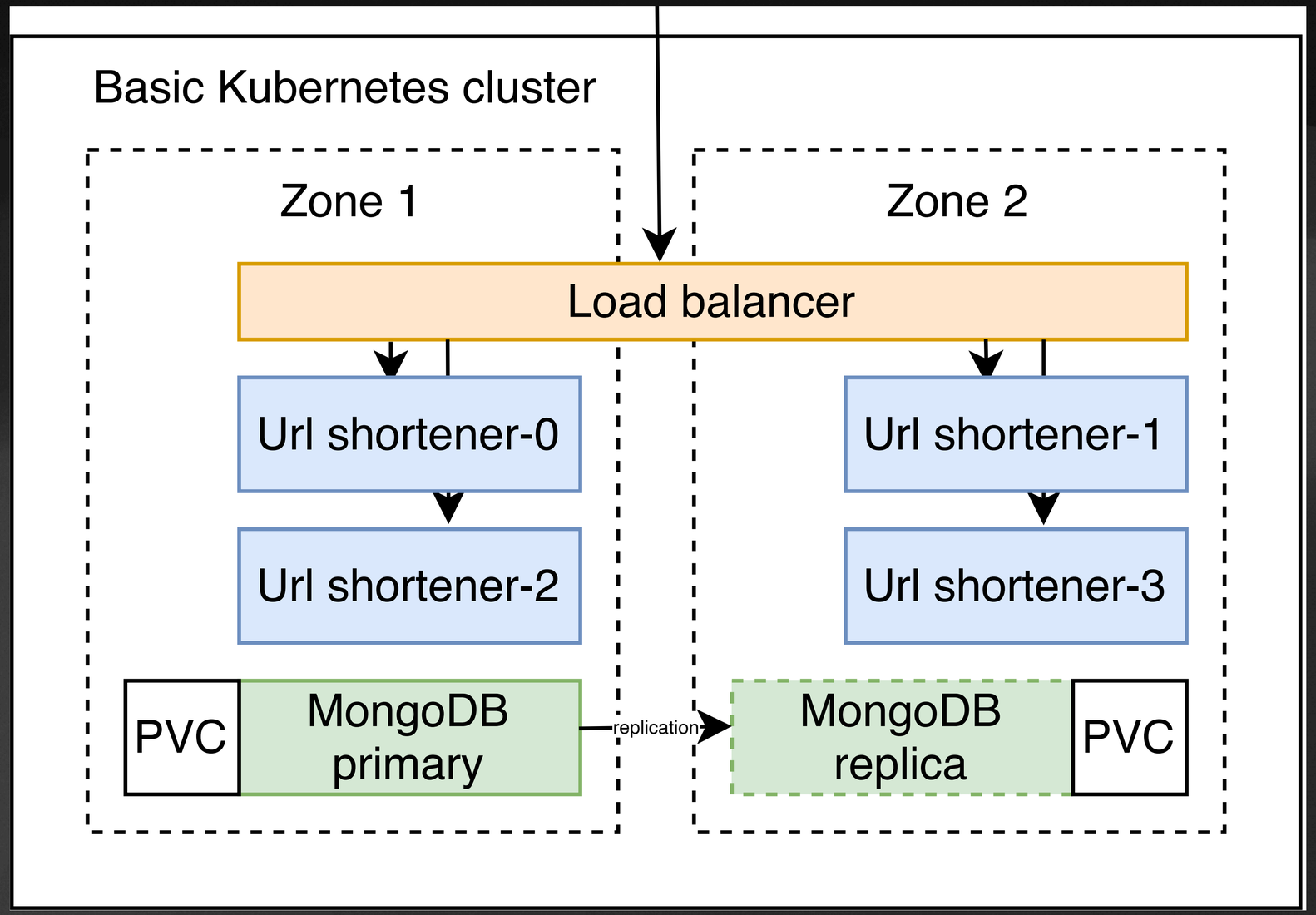
Kubernetes architecture

- At the moment, the application runs locally on a single machine.
- However, thanks to the Dockerfile, the docker image can be deployed on any machine.
- Thanks to the application.properties, the application and the database can run separately

```
spring.data.mongodb.host=${DB_URL}
spring.data.mongodb.port=27017
spring.data.mongodb.database=urlShortener
spring.data.mongodb.username=${DB_USER}
spring.data.mongodb.password=${DB_PASSWORD}
```

- As such, we can think of a potential Kubernetes architecture to fulfil the high availability requirement.

Basic kubernetes cluster architecture



Considerations

- The url shortener pods could be controlled by either a stateful set or a deployment, with node affinity configured so that pods are scattered between the zones.
- MongoDB could be configured with a replica set, with a pod on each zone. In this configuration, one of the pods will be selected as the primary, and will handle all of the writing traffic. The other one would be eventually consistent.
- To handle some values like the password for the connection to MongoDB, secrets could be used

Advanced architecture

- A front end application could be added to the cluster.
- Prometheus and Grafana could be configured for monitoring (actuator dependency on the backend, prometheus bridge on the cluster).
- Logs could be forwarded to Splunk using the Otel collector.
- Backups could also be added to the cluster, using velero
- Multiple independent clusters could be configured, to have different environments (DEV, INT, PROD…)
- An external identity provider for OAUTH instead of basic auth could be considered.

Possible CI and CD implementations

CI

- Run application tests
- Generate a new tag with the new version number
- Build a new version of the url shortener (docker build…)
- Push the new version to the application registry
- Push the new tag to the version control provider (GitHub, GitLab, Bitbucket…)

Possible CI and CD implementations

CD

- Save the tag of the current version of the url shortener deployed on desired the cluster.
- Pull the specified image from the application registry.
- Build the image in the desired cluster of the application, and wait until all pods are running.
- If the pods present an error or do not initialise, roll the deployment back to the previous tag and fail the pipeline.

Questions