Microservices: Architecture, Design and Implementation

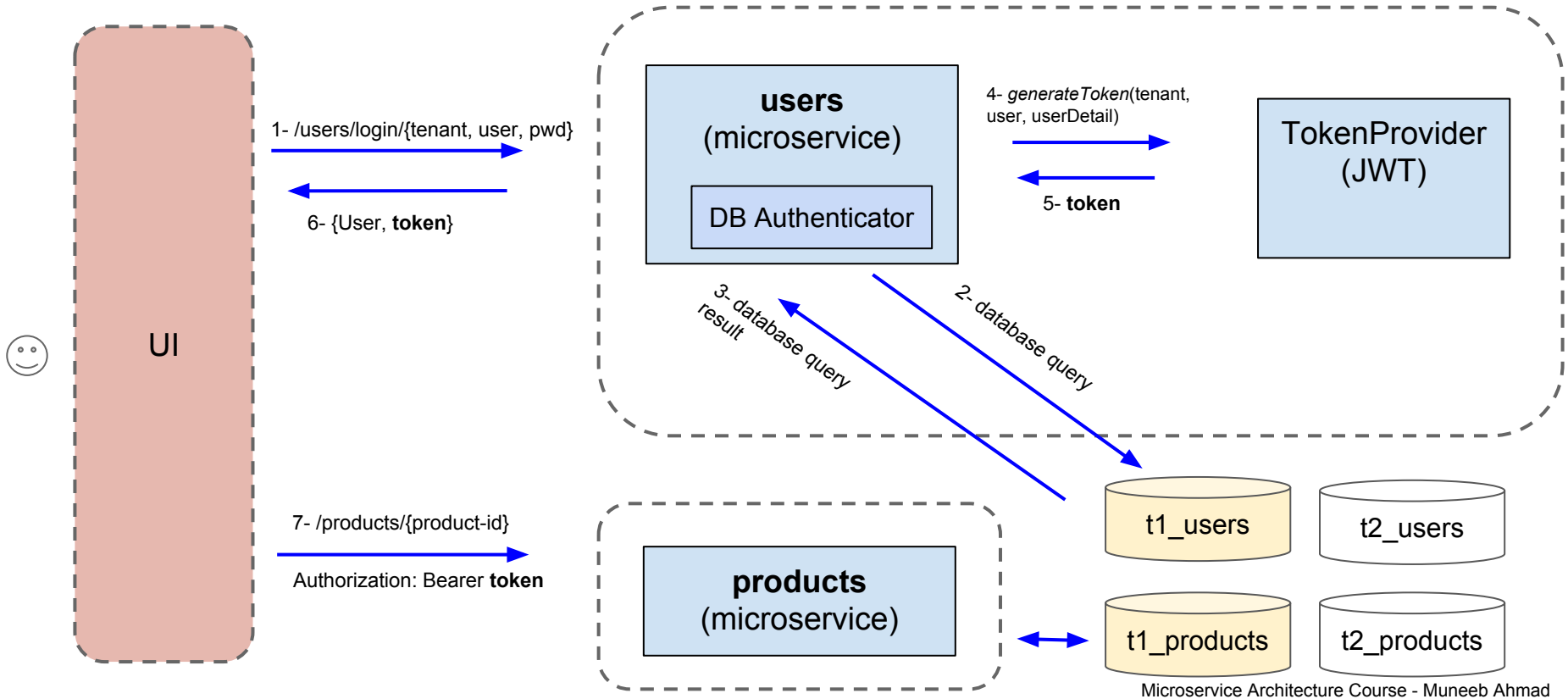
(Patterns and Best Practices)

Muneeb Ahmad

Agenda: Week 5 (of 5)

- Review of last week’s work/labs
- Week 5 Labs
- Misc Topics
  - AWS Infrastructure
  - Deployment Environments
  - Architecture to support Multitenancy
  - Hybrid Architecture
  - Feature Toggle
  - Microservices Patterns
  - API Versioning

Data Flow: Users/Products services



(Last week’s review)

Microservice: jshop-ms-products

(from specs to implementation)

REST API Specs

https://docs.google.com/document/d/1o4fpMNPzgeOz7u8ZxEcxvZg1qWN75DZZz8eI0Y0OdQw/edit

Data Persistence Specs

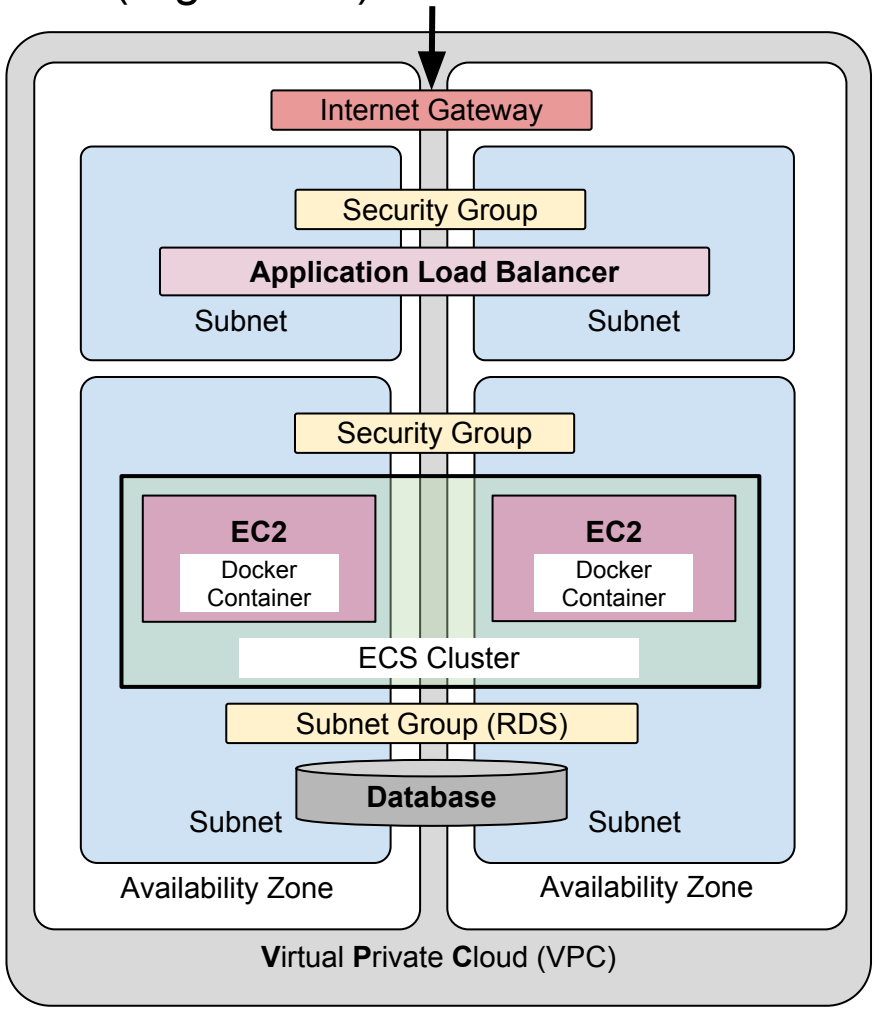
https://docs.google.com/document/d/1rnA4EfjgZ5Rd38yP84no-Iinqh0wC3nE_6HVQwLdk6w/edit

LAB

Products Microservice using Enterprise App structure

https://docs.google.com/document/d/1-2v5C_ezU7h6bQDz5zmla-JpzkIS2erQr52wUwwK-E4/edit

AWS Infrastructure (High Level)



Labs

- Lab 5.1 Build Base Image - Optional
  - https://drive.google.com/open?id=1MnaADSuB2sqfEZNrvptBebpDoxPYCUuPvsMUAdYUoYI
- Lab 5.2 Build and Push Docker Image - Products service
  - https://drive.google.com/open?id=12EdXBAdfSkKf-HkVCrZqs7O5ae8XA-6pFV1JqT8Ibx8
- Lab 5.3 Setup Database on AWS
  - https://drive.google.com/open?id=1-97VuG_HqERQKtnz02O4TRulqMDJ06EvJU_SY5U5F-U
- Lab 5.4 Deploy Products Service to AWS ECS
  - https://drive.google.com/open?id=15SWqrX7cy2QiHnpAobFy1mnb3_9IFfv2X4XGYJ3D6gg

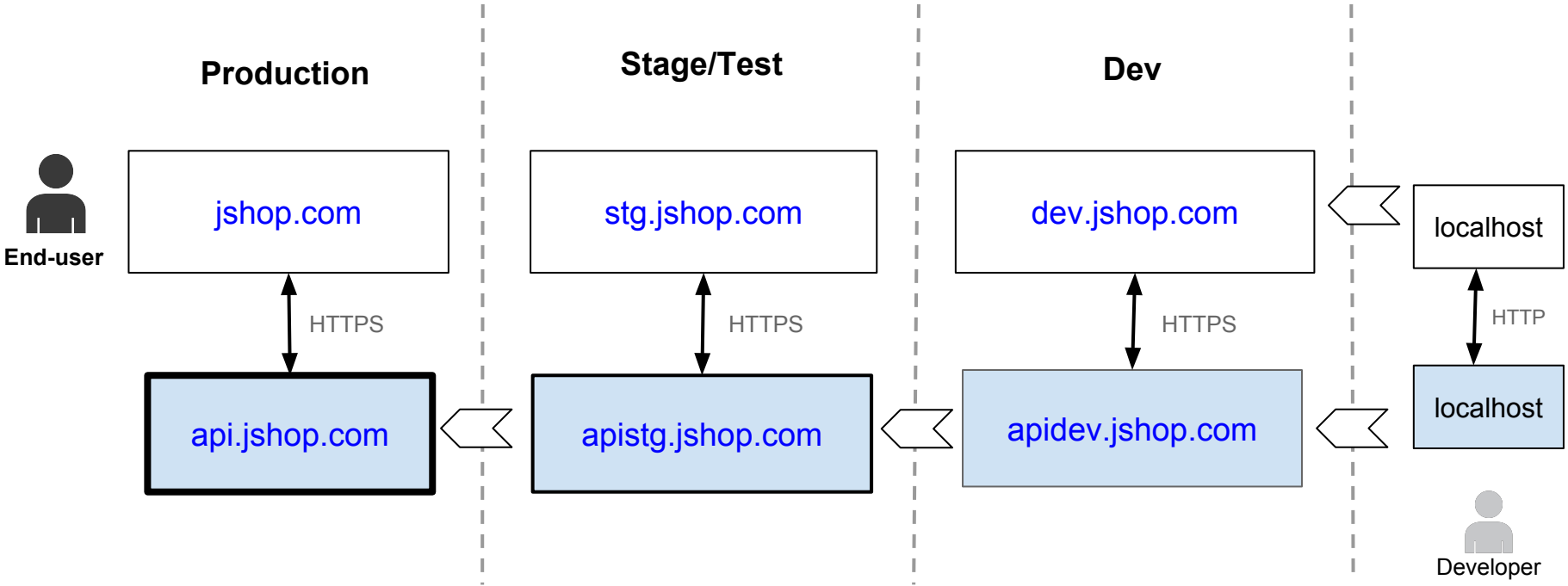
Course Contents

- **Microservices Architecture**
- **Microservice Design**
- **Tools: Git, IDE, Spring, Docker**
- **Projects and Modules**
- **Build and Deploy**
- **Development Lifecycle**
- **AWS Container Service**
- Infrastructure Automation
- **Deployment Environments**
- **Microservices Patterns**
- **Multi-tenancy**
- **Feature Toggle**
- **12-factor app**
- **API Versioning**
- **Hybrid Architecture**

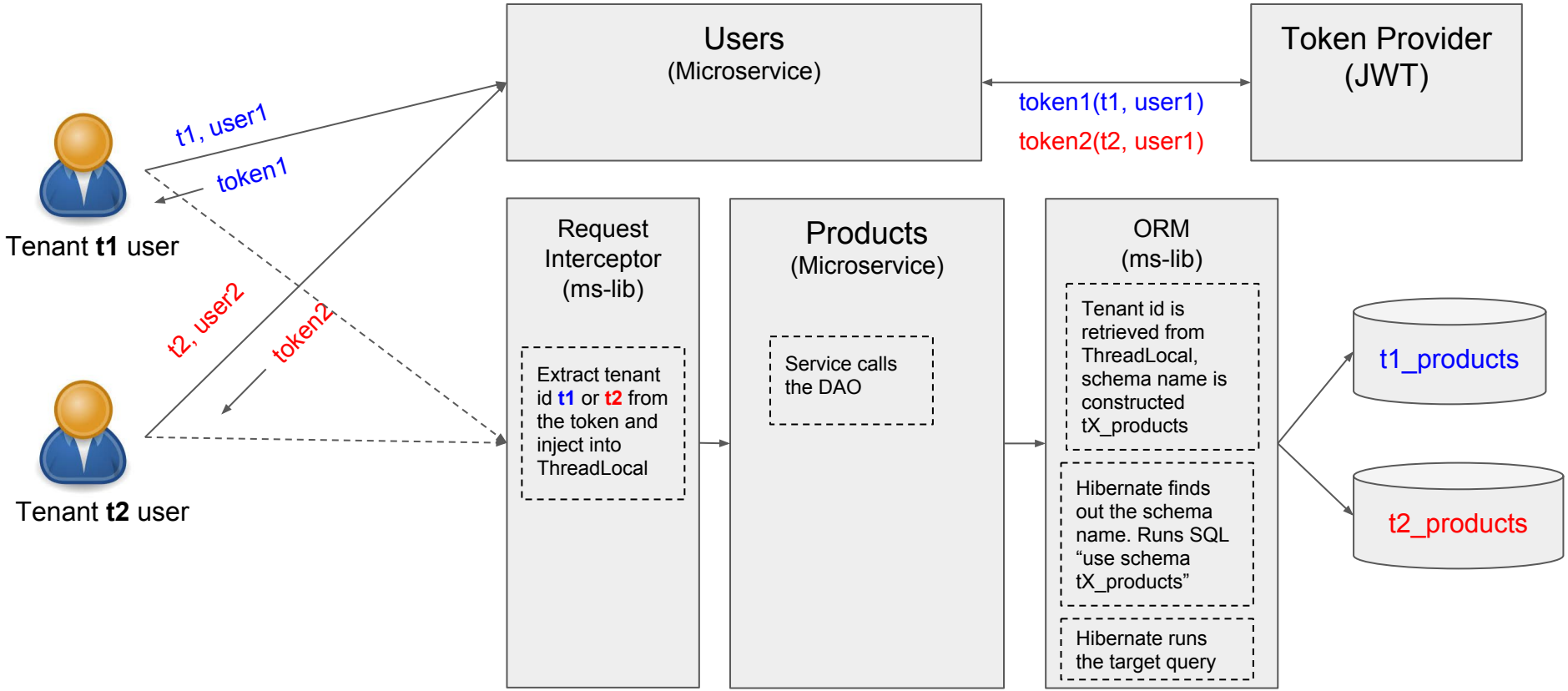
Misc. Questions/Answers

- How do I set up dynamic port mapping for Amazon ECS?
  - https://aws.amazon.com/premiumsupport/knowledge-center/dynamic-port-mapping-ecs/
- How to Unit Test GUI code?
  - https://www.theregister.co.uk/2007/10/22/gui_unit_testing/
- How to Refactor a Monolith into Microservices?
  - https://www.nginx.com/blog/refactoring-a-monolith-into-microservices/

Deployment Environments

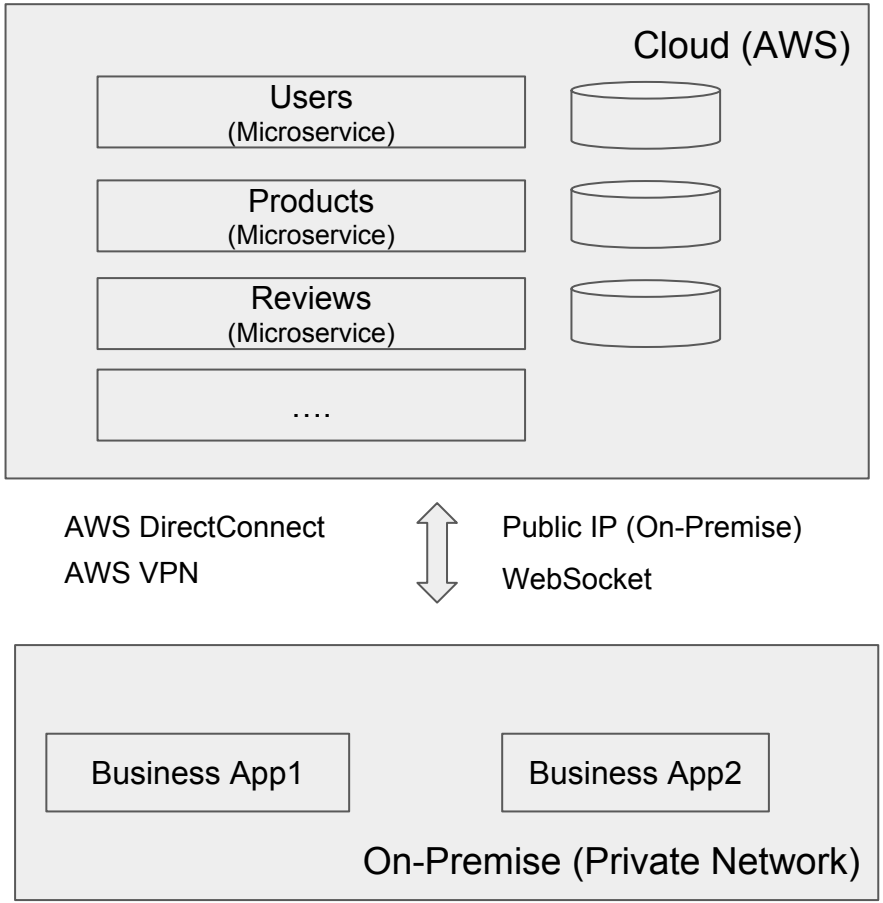


Multitenancy (Data Access)



Reference: https://docs.jboss.org/hibernate/core/4.2/devguide/en-US/html/ch16.html

Hybrid Architecture



Feature Toggle

Backend Service

```
GET http://jshop.com//rest/tenants/enabled-features/{tenant}
Accept: application/json
Authorization: Bearer gHtfwQ..raiFthRW

200 OK
{
  "enabledFeatures" : [ "feature1", "feature2", "feature2", .....]
}
```

Frontend Strategy

1. Create a JavaScript Service to make the REST call to the backend.
2. Create custom tag/directive like f**eature-toggle**,
3. Add the directive to a section you want to show or hide according to the feature status.

```
<div feature-toggle="featureName" action="hide/show">
```

4. The feature-toggle directive may be applied at the page level, menu level or field level.

Microservices Patterns

- Authentication Token
- API Gateway
- Messaging
- Service Discovery
- Health Check
- Schema per Service
- Distributed Logging

Microservices = **SOA**

- ESB -SOAP - Central Persistence - Vendors

+ REST/HTTP + CI/CD + DevOps + Containers + PaaS

Reference: https://twitter.com/arungupta/status/603714264587722754

API Versioning

- Using API path

```
GET http://jshop.com/rest/products/v1
```

- Using Query Parameter

```
GET http://jshop.com/rest/products?version=v1
```

- Using Request Header

```
GET http://jshop.com/rest/products
Accepts-version: v1
```