NetApp

Develop an Extendable REST API server and Composite UI Based on Microservices

Pitt Capstone Project Fall 2019

NetApp Software-Defined Core Infrastructure
September 13, 2019

Agenda

1) Introductions
   - Name
   - Interests
   - Previous experience

2) A little about NetApp
   - What do we do?
   - What is Software Defined Core Infrastructure?

3) The Project

4) Introduction to concepts
   - REST
   - Containers & Docker

5) Scheduling
   - Working together with us

About NetApp

- Data Authority in the Hybrid Cloud
  - Software, systems, and services for data
  - Deploying and building providers IT environment
- $6 billion+ in revenue
- Fortune 500® Company
- More than 10,000 employees in 150 offices worldwide
- Our office: PTC – Pittsburgh Technical Center
  - Acquired Spinnaker Networks in 2003
  - 200 Employees – almost all engineering

Introductions

NetApp Engineers

Pitt Students

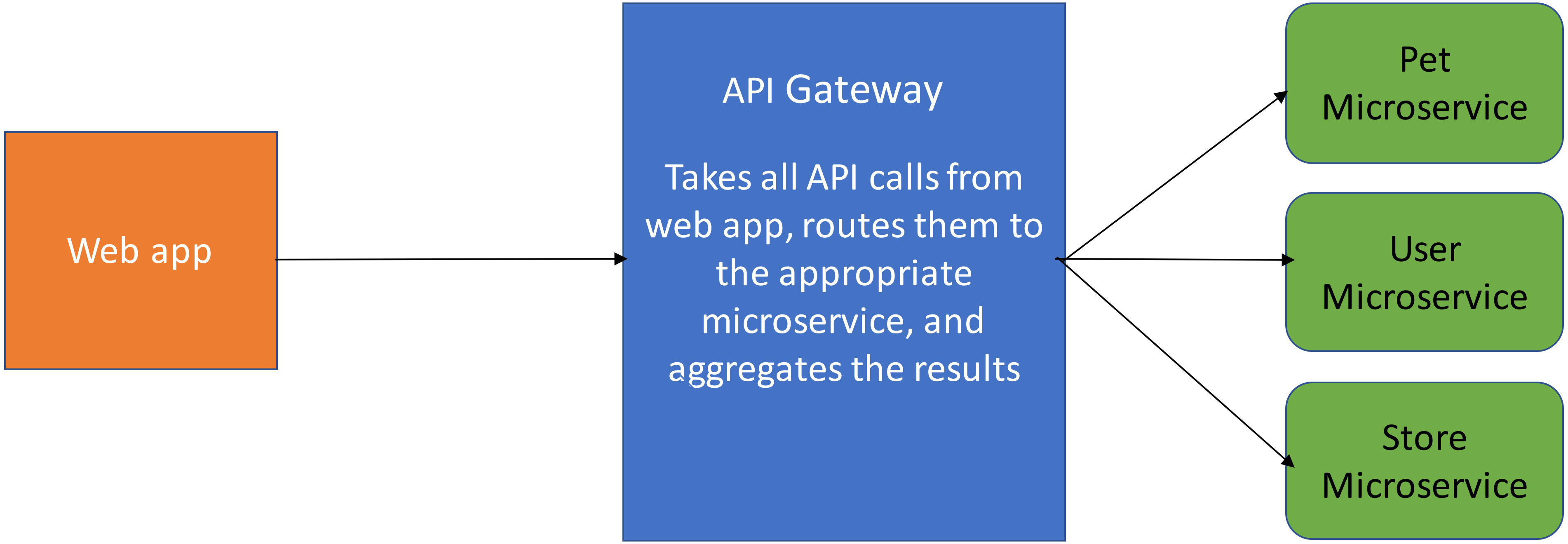
Our Backgrounds

Where we were before NetApp

- **Charlie Mietzner** (Charlie.Mietzner@netapp.com)
  - Graduated in Fall 2016 from Pitt with BS in CS and BA in English Literature
  - Capstone Project for NetApp in Fall 2016
  - Started at NetApp in mid-July 2017
- **Twesha Mitra** (Twesha.Mitra@netapp.com)
  - Graduated in Spring 2018 with BS in Economics and BS in CS.
  - Full time at NetApp since June 2018

The Project

A proof-of-concept Web application built on microservices and an API gateway


Web app
API Gateway
Takes all API calls from web app, routes them to the appropriate microservice, and aggregates the results
Pet Microservice
User Microservice
Store Microservice

Why a petstore?

It's not about the destination; it's about the journey.

- A Web Application
  for the theoretical owner and employees of a pet store.
- Keeps track of pets, customers, and customer orders in the store.
- Under the covers it must be implemented with a microservice architecture

- An investigative and experimental project
- There isn’t a "right answer"
- Future groups can model their work after your architecture
- We are equally interested in the pitfalls encountered

Deliverables

A proof-of-concept Web application built on microservices and an API gateway

Deliverables/Requirements:

- Workflows should be working as expected
- All endpoints/actions not covered in workflows should also be supported
- Web GUI (client app) that implements the workflows

Workflow 1:

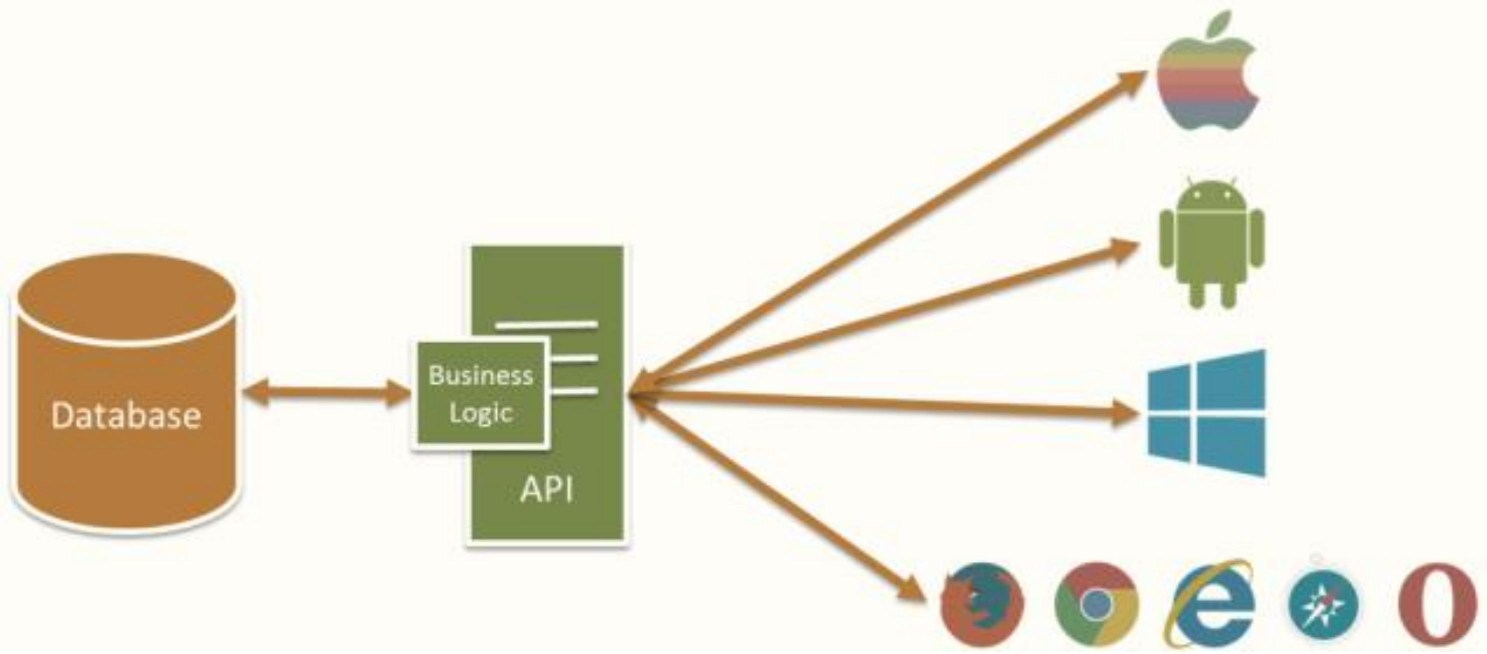
- Register a customer
- View all available pets
- Customer buys a pet
- Owner creates a store order for the pet
- Owner updates pet/user info

Workflow 2:

- Existing customer wants to update their contact information
- Customer has a pet of type A, want to buy another pet of type A
- Create store order for the pet and update relevant information

What is REST (REpresentational State Transfer)

- An application should interact with a resource with only the following knowledge:
  - Resource identifier (URLs, also known as endpoint)
  - Action (GET, POST, PUT, PATCH, DELETE)
  - Understanding of the representation returned (JSON, XML, HTML)



REST constraints

- Stateless
  - Each request from client to server must contain all information necessary to understand the request
  - Scalable and distributable
- Cacheable
  - Resources can be cached, but they must declare themselves cacheable
- Layered system
  - Each component cannot see beyond the immediate layer with which they are interacting
- Uniform interface
  - Uniform way of interacting with a server independent of device/application
  - Identification & manipulation
- Client-server separation
  - Client application and server application should be able to evolve separately without any dependencies on each other

Containers?

Provides a similar benefit of a Virtual Machine

- A virtual machine uses emulated hardware to run a guest OS; a container is effectively an emulated OS.
- Relative to a VM, containers provide isolation and portability in a more lightweight...well...container

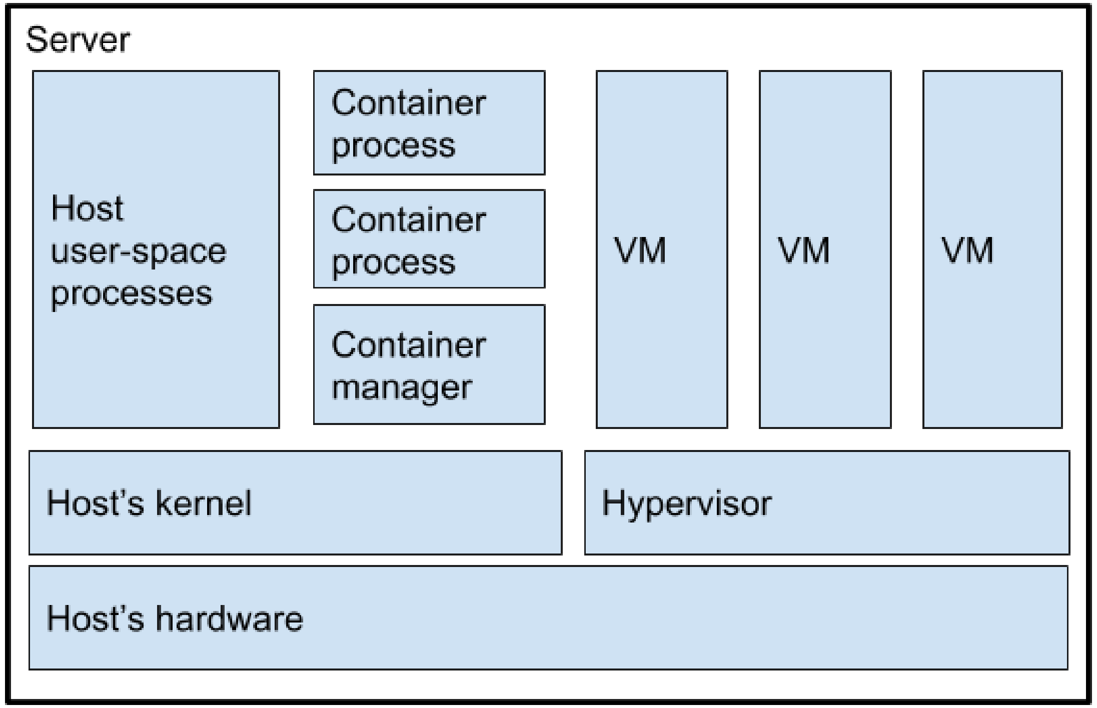


Figure 1. VM vs Container Comparison. Adapted from "What is the difference between a process, a container, and a VM?" By Jessica G. *Medium*. 2019. Retrieved from https://medium.com/@jessgreb01/what-is-the-difference-between-a-process-a-container-and-a-vm-f36ba0f8a8f7

Microservices?

A Software Architectural Style

- A software application composed of independently deployable services.
- Think "highly cohesive, loosely coupled".
- Each piece of the application can be developed and deployed independently.
- Each microservice within the application can use different technology stacks, languages, communication protocols—it just needs to satisfy a well-defined API
- Fault isolation. A bug in one microservice doesn't *technically* affect other microservices in the application

API Gateway?

Why do we need one?

- Lots of microservices requires that the client keep track of multiple hosts / service instances.
- Each microservice (by design) could have different interfaces (HTTP, AMQP, RPC, etc...). The client would need to switch between these to interact with the full application.
- Interacting with several microservices requires the client to make several network requests.
- The client would need to implement their own solutions to achieve complicated application-level workflows requiring several microservices.

What should it provide?

- Think big-picture workflows.
- Think about the needs of the consumer of your application.
- There doesn't need to be just *one* API gateway. E.g. Netflix

Scheduling

- Meet once a week with entire team
  - In person every other week
  - Over the phone every other week
- Midterm presentation
- Final Presentation
- Communication
  - Email, Slack
- Progress tracking and source control
  - Github
  - Creating a project board on github

Where to get started?

- Populate kanban board on github with tasks below
- Schedule midterm/final presentations
- Decide on weekly meeting time & place
- Get a basic REST server up and running (HTTP) in a Docker container
- Come up with a system design
    - Roadmap on how to implement
    - Technologies? Languages?
    - Populate product backlog in github