Principles of Software Construction: Objects, Design, and Concurrency

DevOps

Michael Hilton Bogdan Vasilescu

Carnegie Mellon University
School of Computer Science

institute for SOFTWARE RESEARCH

Administrivia

- Final Exam: Monday, May 6, 2019 05:30 p.m. - 08:30 p.m.
  - **LOCATION: GHC 4401**
  - Review Session Saturday, May 4th, 1-3pm in NSH 3305

COURSE DESCRIPTION

The course takes a software engineering perspective on building software systems with a significant machine learning or AI component. It discusses how to take an idea and a model developed by a data scientist (e.g., scripts and Jupyter notebook) and deploy it as part of scalable and maintainable system (e.g., mobile apps, web applications, IoT devices). Rather than focusing on modeling and learning itself, this course assumes a working relationship with a data scientist and focuses on issues of design, implementation, operation, and assurance and how those interact with the data scientist's modeling.

This course is aimed at software engineers who want to understand the specific challenges of working with AI components and at data scientists who want to understand the challenges of getting a prototype model into production; it facilitates communication and collaboration between both roles.

WHAT QUESTIONS WILL THIS COURSE ADDRESS?

- How can correctness or usefulness of a system with an AI component be specified or evaluated? How does requirements engineering change for AI-enabled systems?
- How to analyze and mitigate wrong results and how to design robust systems? Is modular design still possible with AI components?
- How and where to deploy models, how and when to update models, and what telemetry to collect? How to design learning and evaluation infrastructure that scales?
- How to compose multiple AI components within a system and detect feedback loops? What does software architecture for AI-enabled systems look like?
- How to detect poor data quality, poor model quality, and data drift? What would unit testing for data look like?
- How to assure quality of an AI-enabled system? How would test automation look like to test correctness of infrastructure or models?
- How to assure fairness and privacy of AI-enabled systems?

LOGISTICS

When: **Monday/Wednesday 1:30-2:50 // FALL 2019**

Instructors: **Christian Kästner**
kaestner@cs.cmu.edu
www.cs.cmu.edu/~ckaestne/

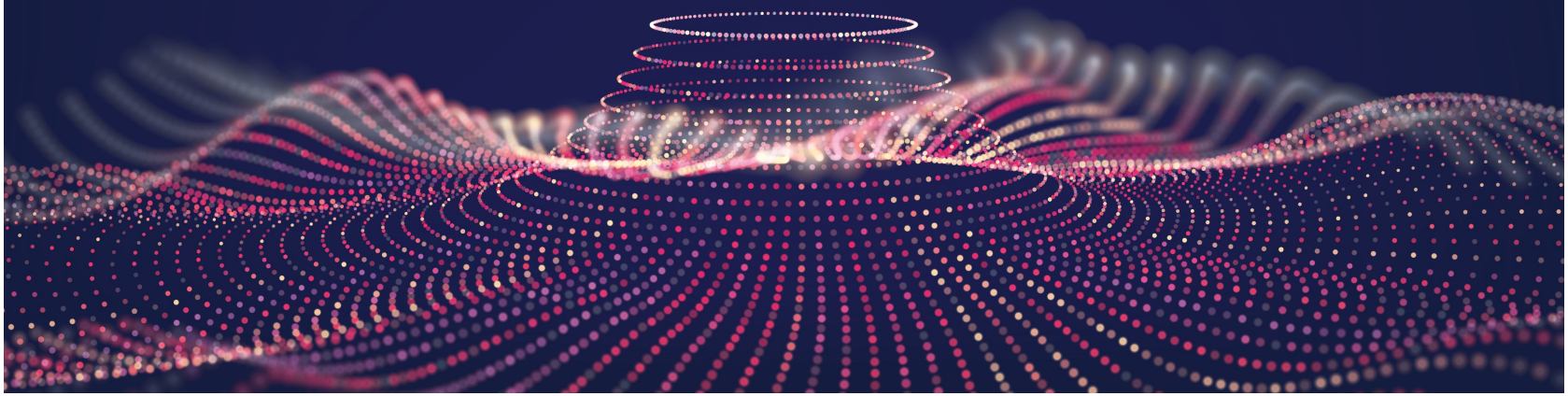
Eunsuk Kang
eunsukk@andrew.cmu.edu
eskang.github.io

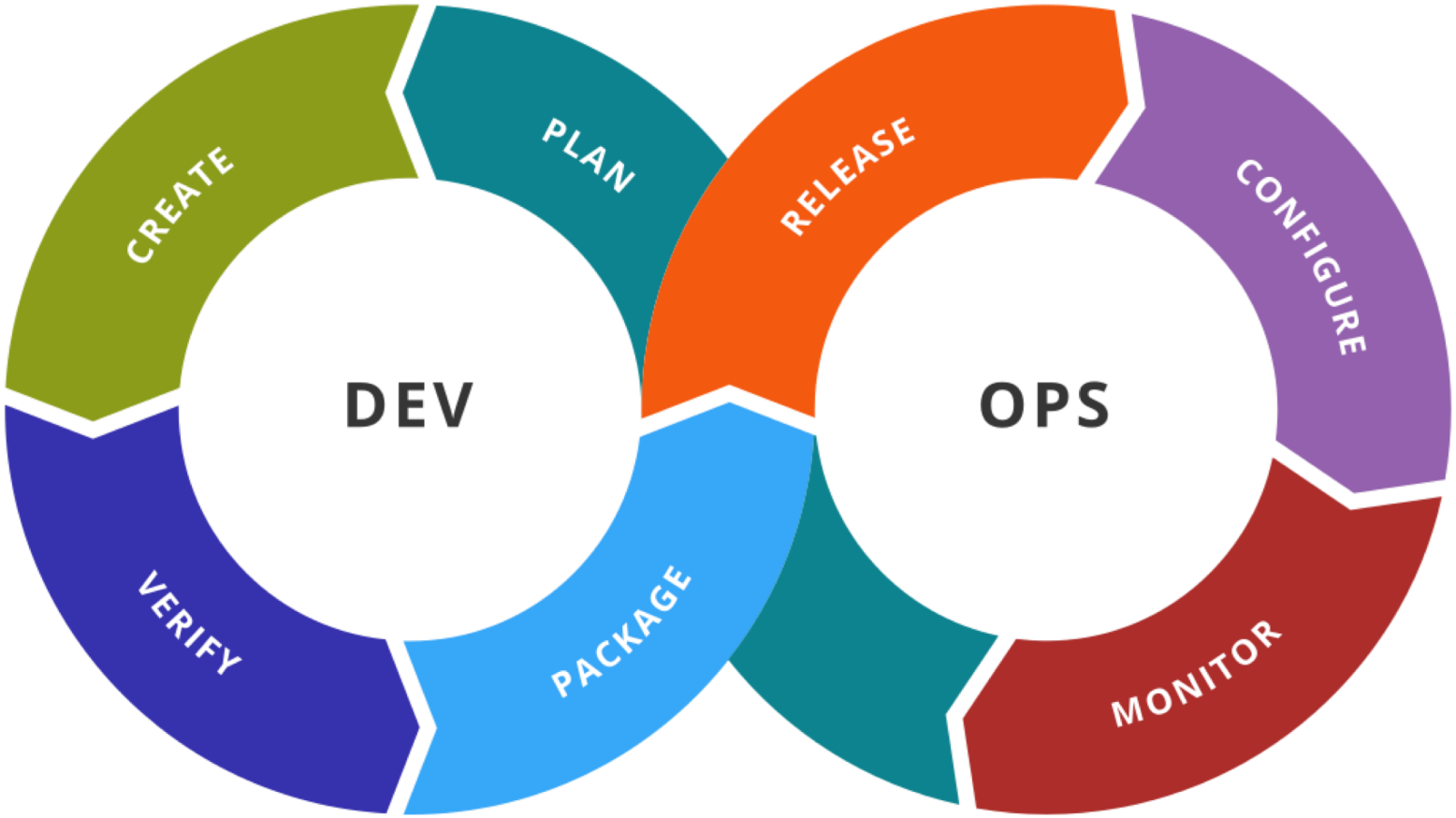
Open to undergraduate and master students meeting the prerequisites.

Visit the course website to learn more:
ckaestne.github.io/seai/

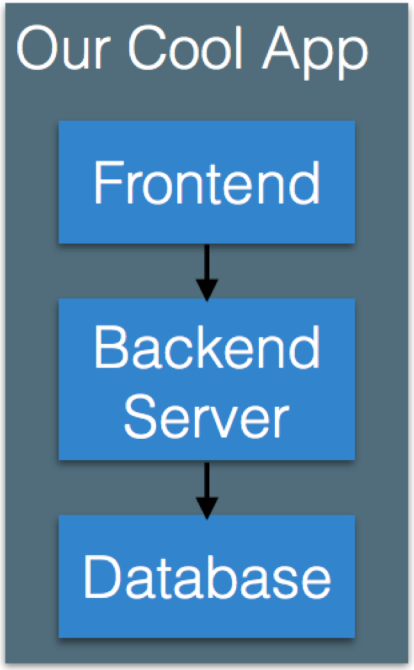
SOFTWARE ENGINEERING FOR AI-ENABLED SYSTEMS

17-445/645 · FALL 2019

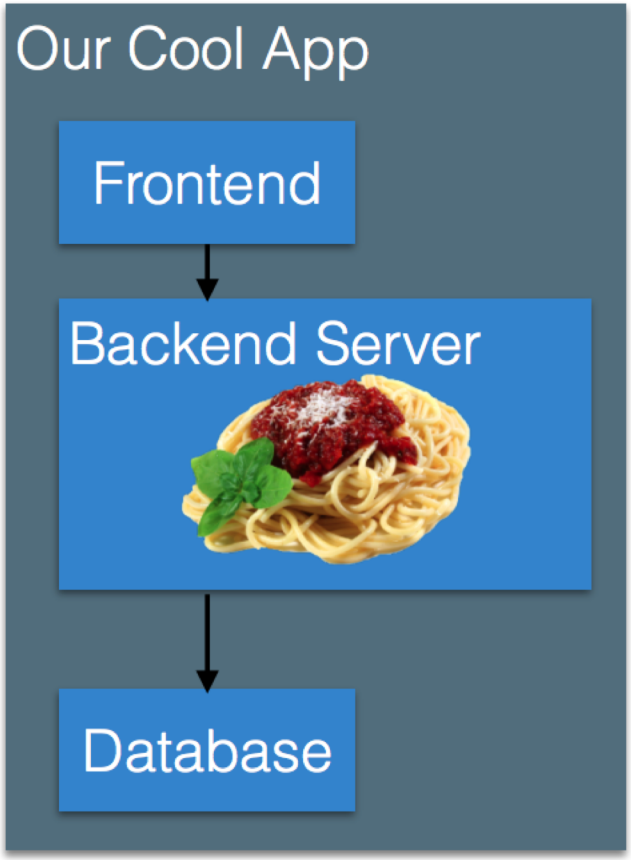



CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

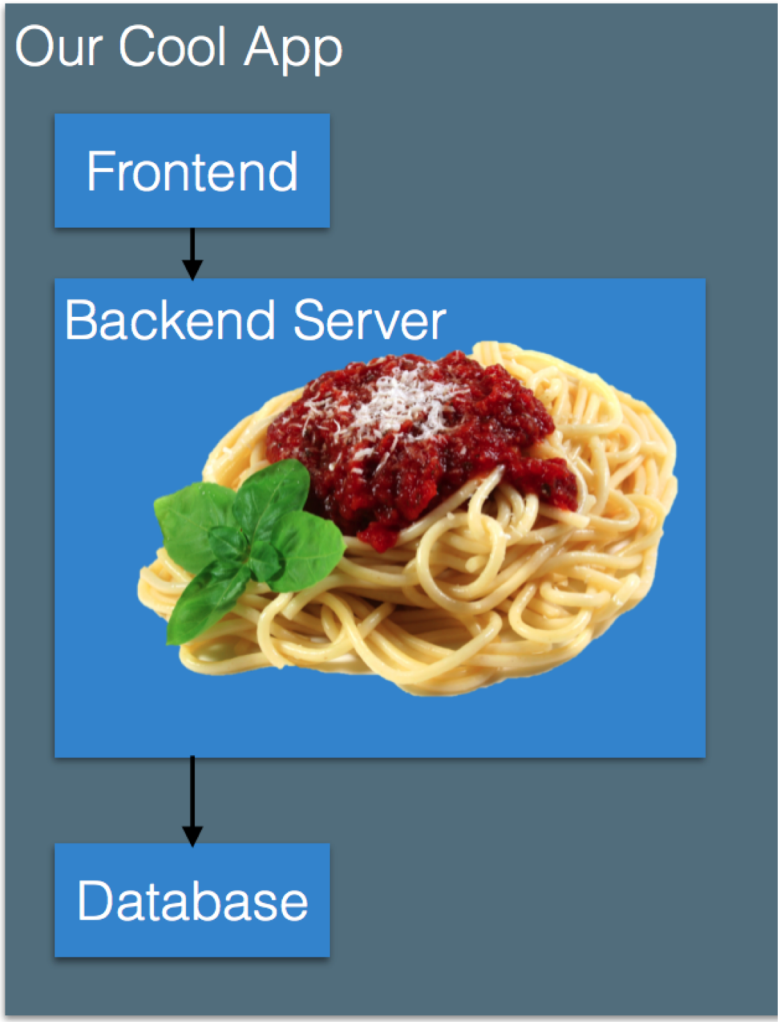
Simple Layers App



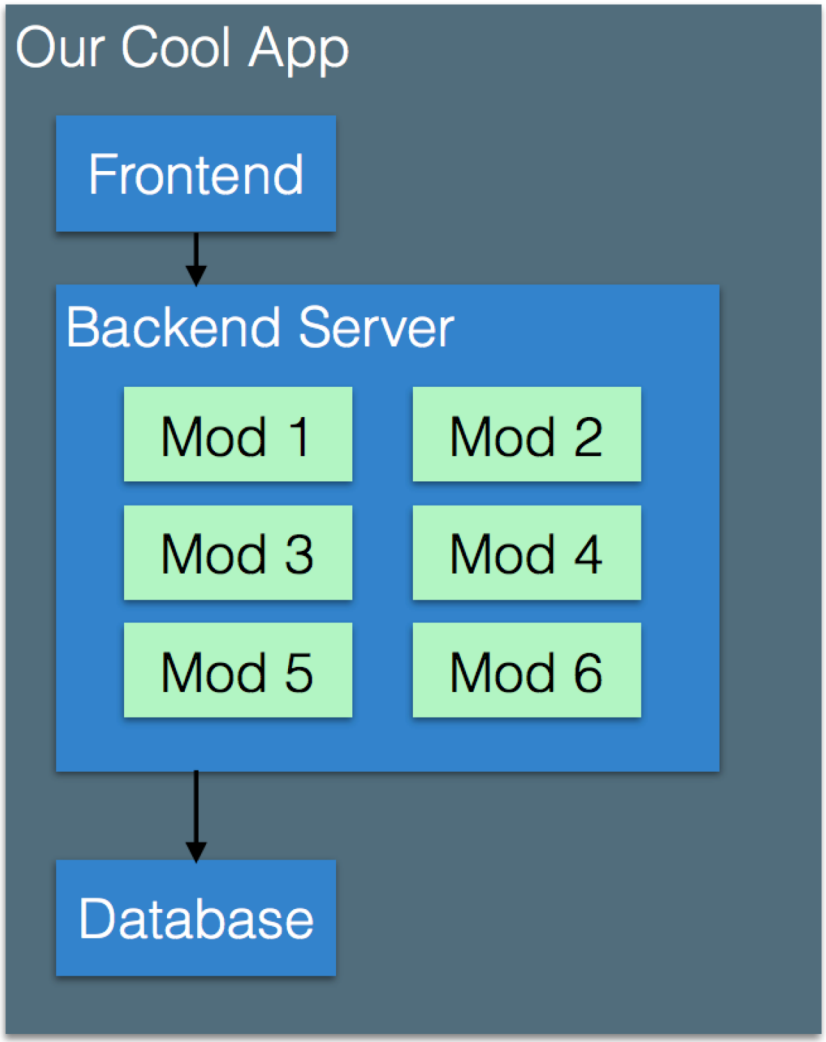
More functionality



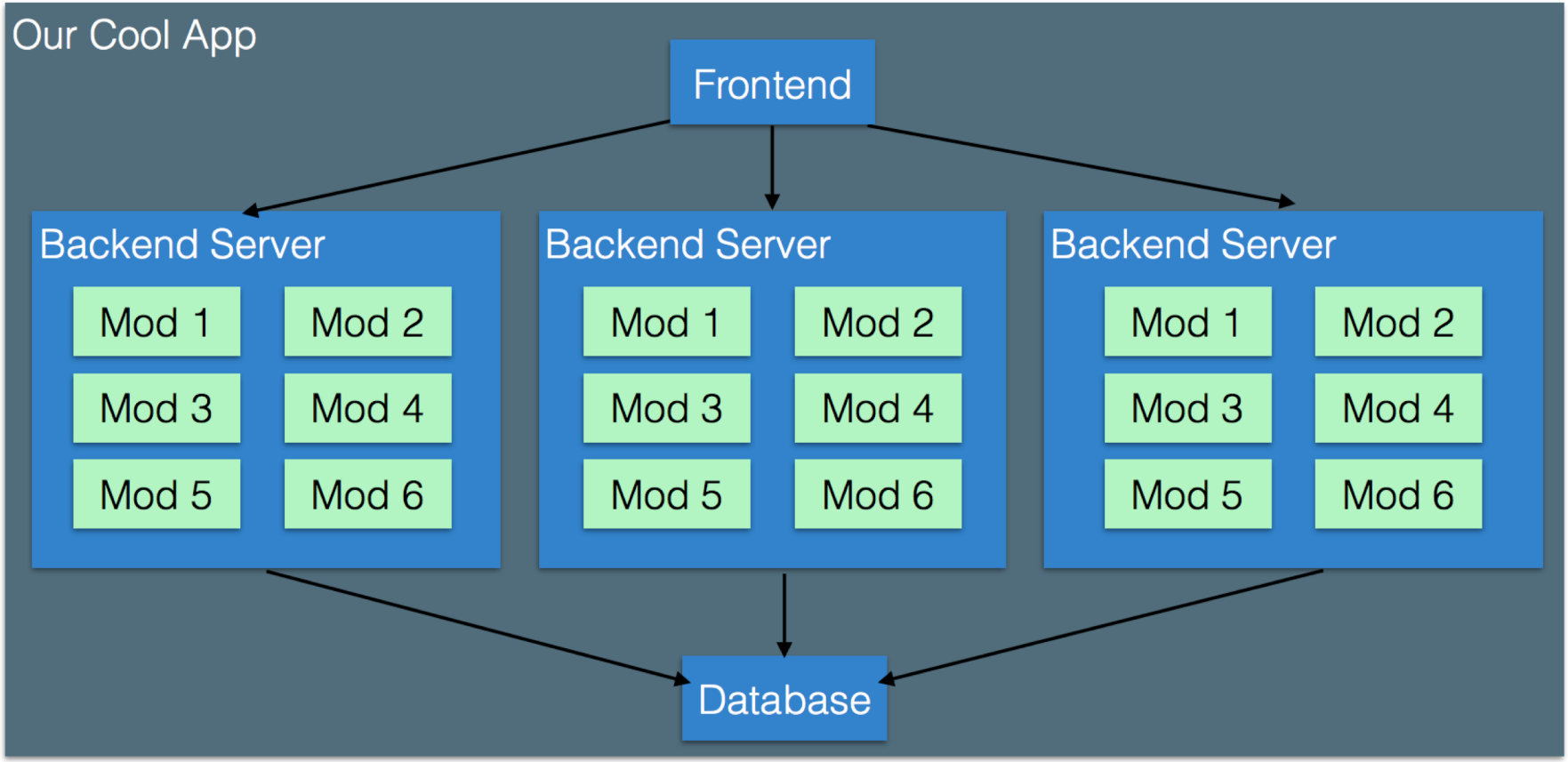
Even more functionality



Organize our backend

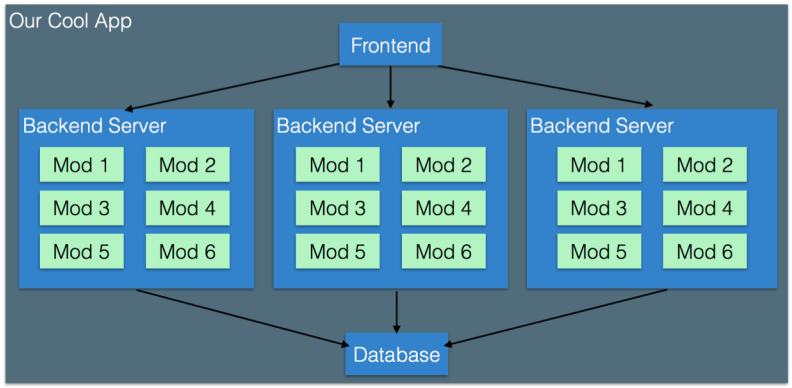


How to scale?

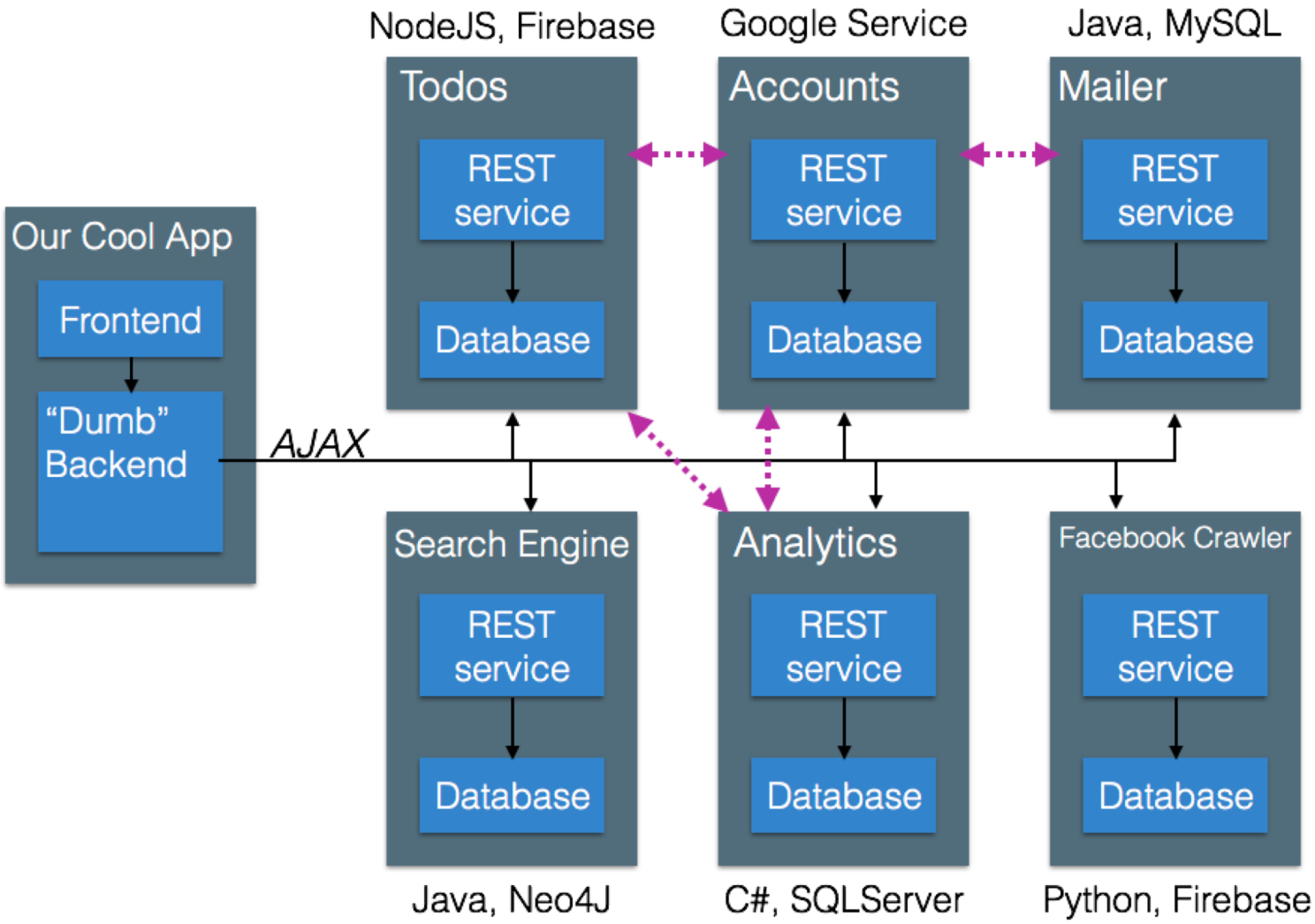

Our Cool App
Frontend
Backend Server
Mod 1
Mod 2
Mod 3
Mod 4
Mod 5
Mod 6
Backend Server
Mod 1
Mod 2
Mod 3
Mod 4
Mod 5
Mod 6
Backend Server
Mod 1
Mod 2
Mod 3
Mod 4
Mod 5
Mod 6
Database

Monolith

- What happens when we need 100 servers?
- What if we don't use all modules equally?
- How can we update individual models?
- Do all modules need to use the same DB, language, runtime, etc?



Microservices

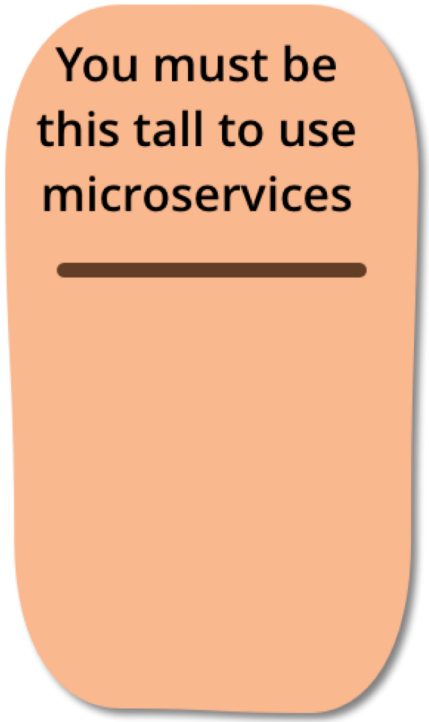


Microservices should be:

- Modelled around business domain
- Culture of automation
- Hide implementation details
- Decentralized governance
- Deploy independently
- Design for failure
- Highly observable

Microservice prerequisites

- Rapid Provisioning
- Basic Monitoring
- Rapid Application Deployment
- Devops Culture




MICROSERVICES...
...SO HOT RIGHT NOW
memegenerator.net

Why are microservices such a big deal?

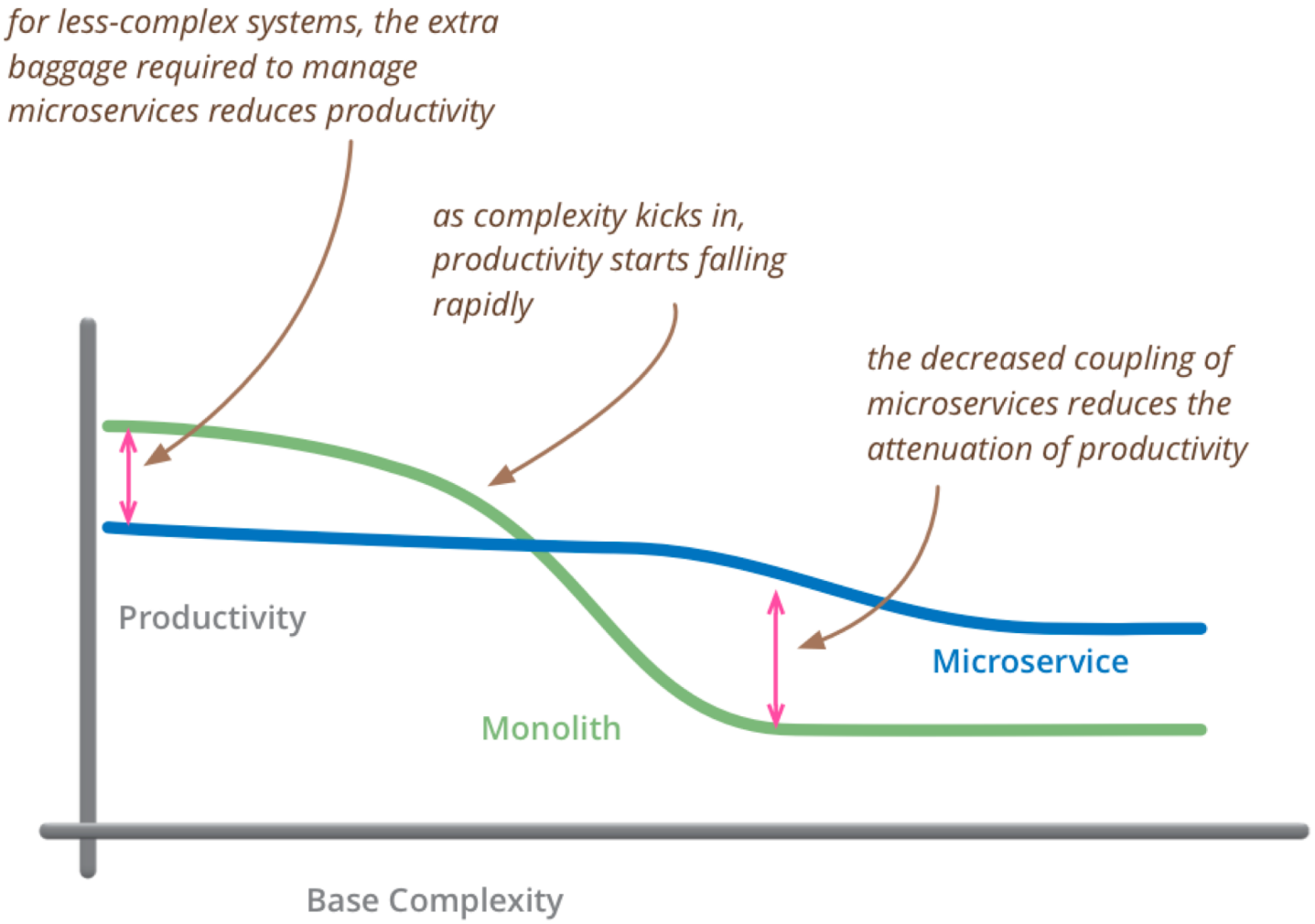
Impact on development practices

- Amazon transitioned to “two-pizza” teams
- “Full Stack” developers
- ”Devops” as a prereq
- Live testing and rollback
- Migrating from “monolith to microservices” is popular, but comes at a cost

Microservices benefits

- Strong Module Boundaries
- Independent Deployment
- Technology Diversity

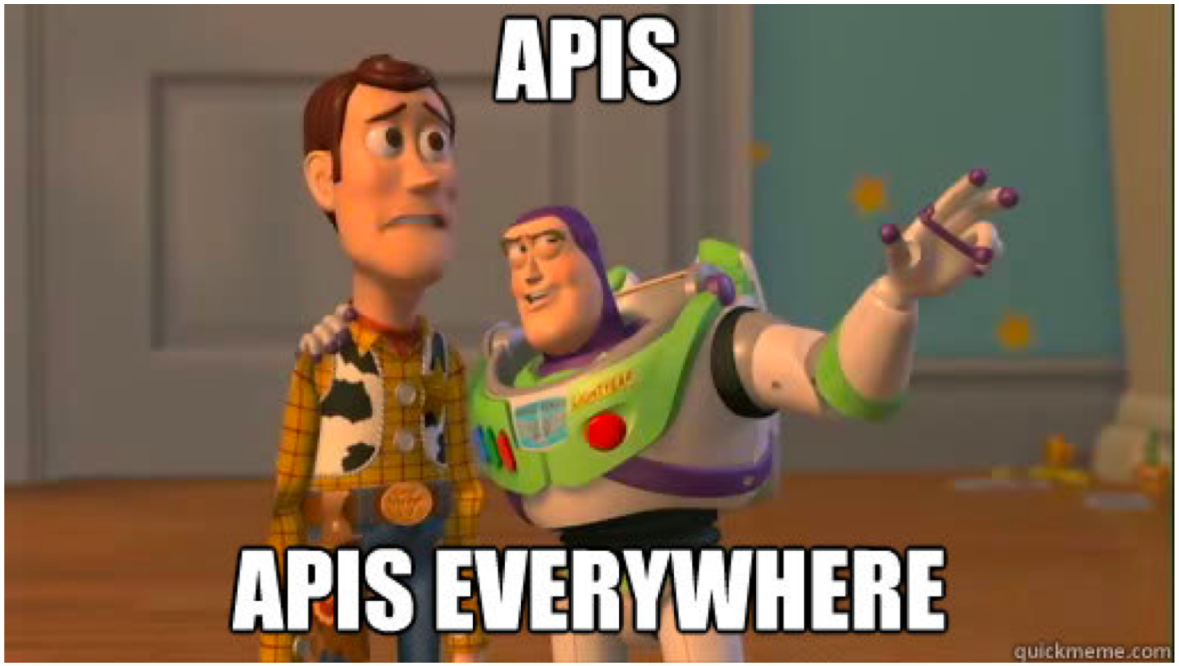
Microservices overhead



but remember the skill of the team will outweigh any monolith/microservice choice

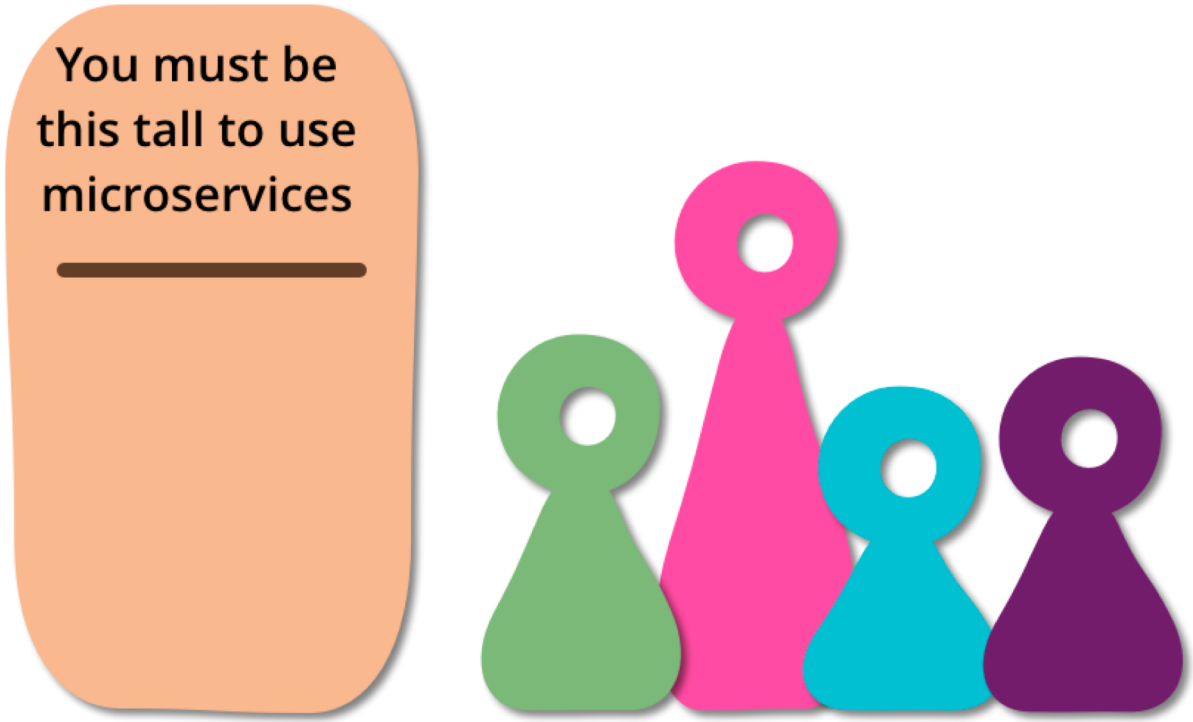
Microservice costs

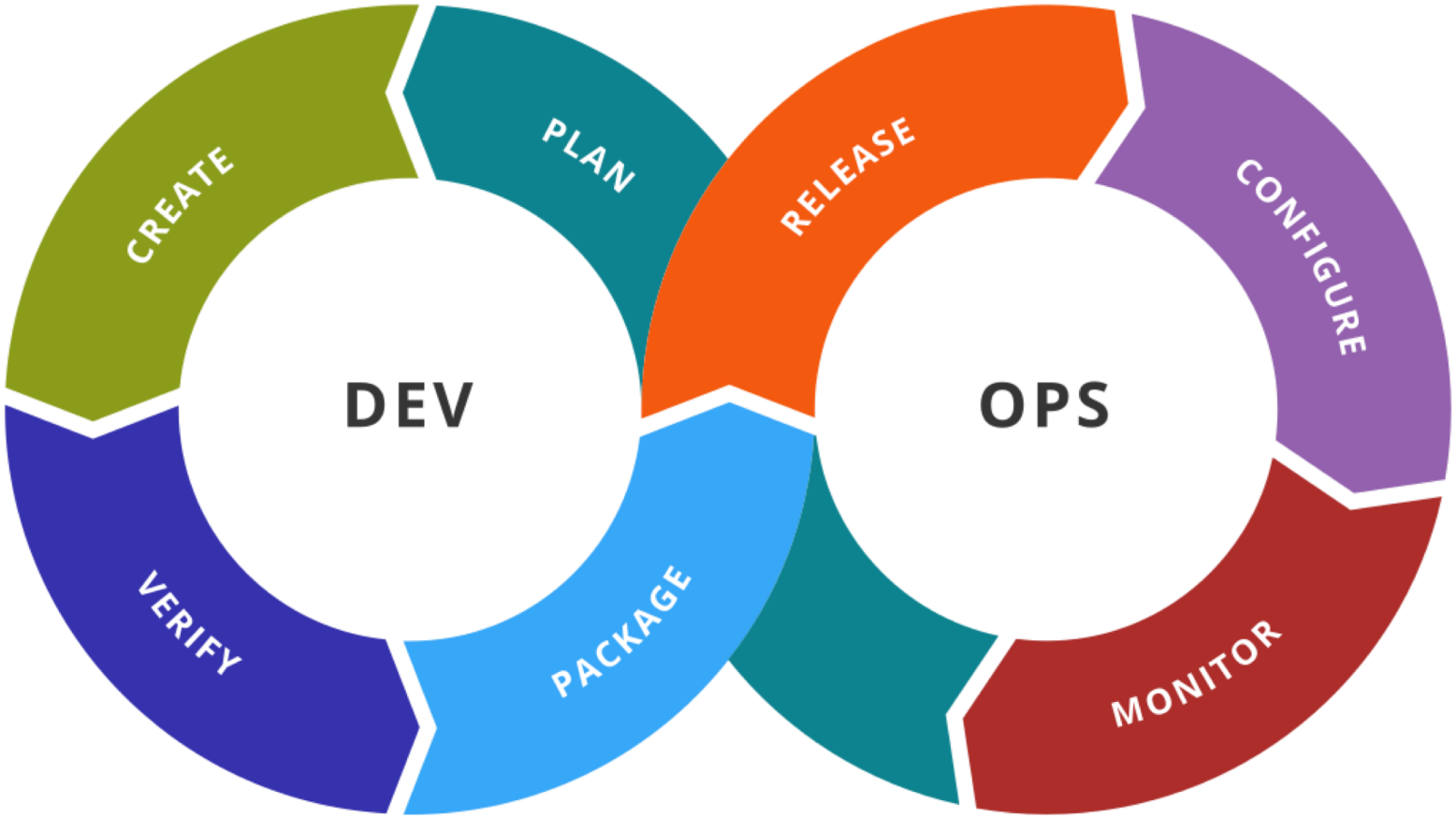
- Distribution
- Eventual Consistency
- Operational complexity
- Leads to more API design decisions



Microservice prerequisites

- Rapid Provisioning
- Basic Monitoring
- Rapid Application Deployment
- **Devops Culture**




CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

Why DevOps?

- Developers and Operations don't have the same goals
  - Devs want to push new features
  - Ops wants to keep the system available (stable, tested, etc.)s
- Poor communication between Dev and Ops
- Limited capacity of operations staff
- Want to reduce time to market for new features
- Reduce "Throw it over the fence" syndrome

DevOps Definition

- "DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality."