
Serverless Academy
Serverless Concepts
Linux Academy Edition

Making the Most of This Course on Linux Academy

- Review the syllabus
- Check out the instructor note cards
- Visit the course community
- Create a course schedule
- Create a study group


Serverless Academy
Serverless Concepts
Course Introduction

About This Course

- Use this as a foundation for future courses
- This course is designed for beginners
- No Labs
- No Cloud Vendor Access Required

Syllabus

- Serverless Definition
  - Backend as a Service
  - Function as a Service
- Comparison of Traditional vs. Serverless Architectures
- Serverless Offerings
  - Cloud Vendors
  - On-Premise Solutions
- Benefits vs. Caveats
- Production Examples
- Ways to Get Started
  - Frameworks


Serverless Academy
Serverless Concepts
What is Serverless?

Are There Really No More Servers?

- No, but the implications for developers are pretty similar
- This question has led to tension
- The community has considered renaming “Serverless”
- You lose the ability to get hold of the server and the command line interface (CLI)

Frustration by Peter Alfred Hess
https://flic.kr/p/5x3DF6

We've Been Here Before

- Datacenter operators felt the pressure
  - Infrastructure as a Service (IaaS) caused a shift in responsibility

- Now developers feel the pressure
  - Serverless is causing a shift in responsibility

Pre-Cloud Responsibilities for Dev and Ops

- Budgeting
- Planning
- Connecting
- Powering
- Housing
- Purchasing

- The responsibility process would start all over again to scale

Infrastructure as a Service (IaaS)

- Delivered dramatic improvements in
  - Cost
  - Agility
  - Scalability
  - Reliability, if architected correctly
- Overall Total Cost of Ownership (TCO) fell
- Pricing can be measured in hours
- Application infrastructure can grow and shrink on usage

Going Cloud Native

- (Mobile) Backend as a Service (BaaS)
  - App developers of mobile and single-page web applications started replacing the monolithic server with a distributed system of loosely-coupled components in the cloud
  - Offloaded need to develop and maintain:
    - Identity
    - Authentication
    - Storage
    - Etc.
  - FrontEnd became a delivery vehicle
  - Time-to-market dropped considerably

The Shift Started

- Focus changed:
  - from applications and servers
  - to tasks and process-flows

- Containerization and micro-services became popular

What Serverless Does Not Mean

- Serverless doesn’t mean there are no servers involved
  - Rather, Serverless means developers don’t have to think about the servers
- You offload responsibility to someone else that you trust:
  - Just like when you offload electricity in your home to someone you trust
- Instead you are plugging into elastic computing services
  - Don’t need to provision resources based on load (like IaaS)
  - Don’t need a lot of planning effort (like On-Premise)
  - Growing the app happens on-demand

Serverless Usage is Growing

- This has some auxiliary benefits:
  - Integration skills can apply to multiple projects
  - Hiring Serverless engineers is easier
    - Familiarity with the Application Programming Interface (API) transfers between jobs


Serverless Academy
Serverless Concepts
The Game Changer

From Serverless BaaS to Serverless FaaS

- Function as a Service (FaaS)
- AWS Lambda is an implementation of FaaS
- Qualities of a FaaS:
  - Execute logic in response to events
    - In this context, all logic (including multiple functions or methods) are grouped into a deployable unit, known as a “Function”
  - Packaging, deployment, scaling all handled transparently
  - Scaling is handled automatically, assuming the function is stateless

FaaS Use Cases

- Scenarios include:
  - Proxy API credentials
  - Database queries
  - Job dispatching
  - Domain logic

- In all of these, the Function is in charge of executing a task or performing some kind of data extract, transform, load processing

- FaaS makes it easy to write and deploy simple micro-services

Similarities to BaaS

- The front-end (mobile or single-page web app) is still:
  - The delivery vehicle
  - Responsible for coordinating and connecting to these BaaS/FaaS services
- Developers do not have access to the servers that host the back-end
- TCO continues to drop dramatically, even more than with IaaS

Differences to BaaS

- FaaS services are usually priced on time of execution and resource consumption (like allocated RAM)
- BaaS services are usually priced on storage

- FaaS services usually spin up for the length of the execution and then clean themselves up
- BaaS services may be long-running, but that's an implementation detail of the service provider

Is PaaS Serverless?

- Serverless includes BaaS and FaaS but what about:
  - Heroku?
  - Openshift?
  - Elastic Beanstalk?

- No, these are Platform as a Service (PaaS) providers

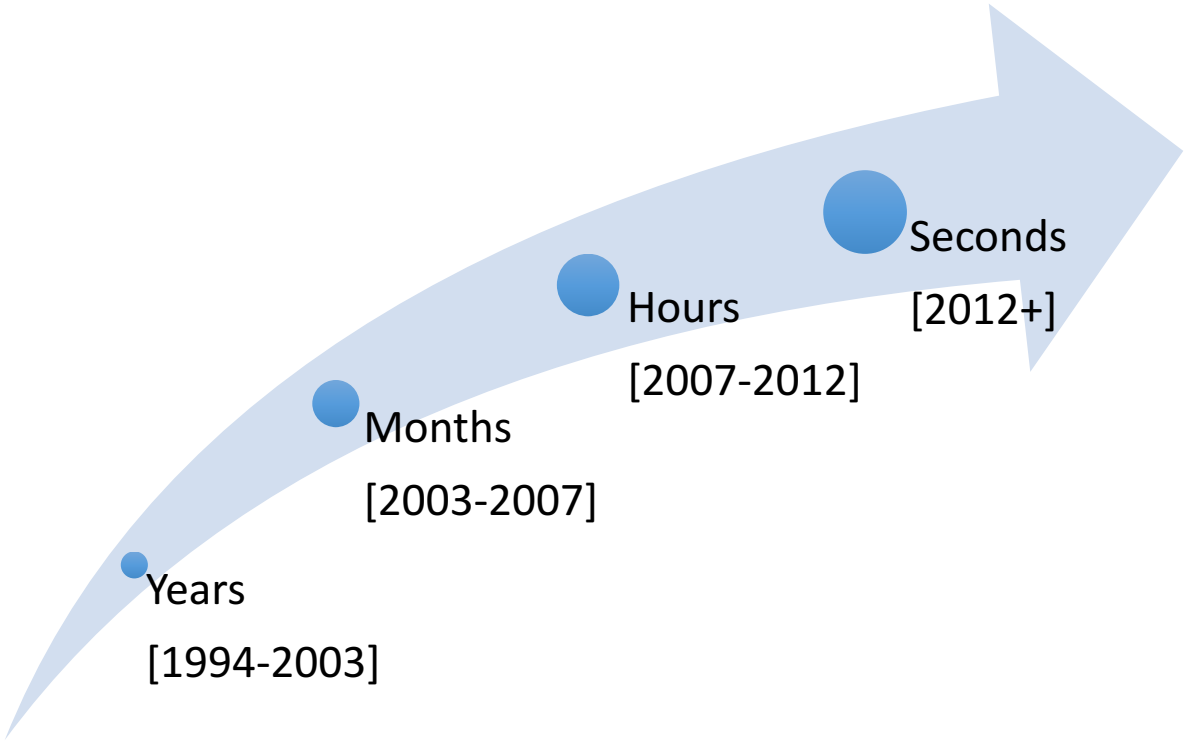
- PaaS still forces the developer to control the number of instances running, this concern is specifically what Serverless removes

Is This Serverless?

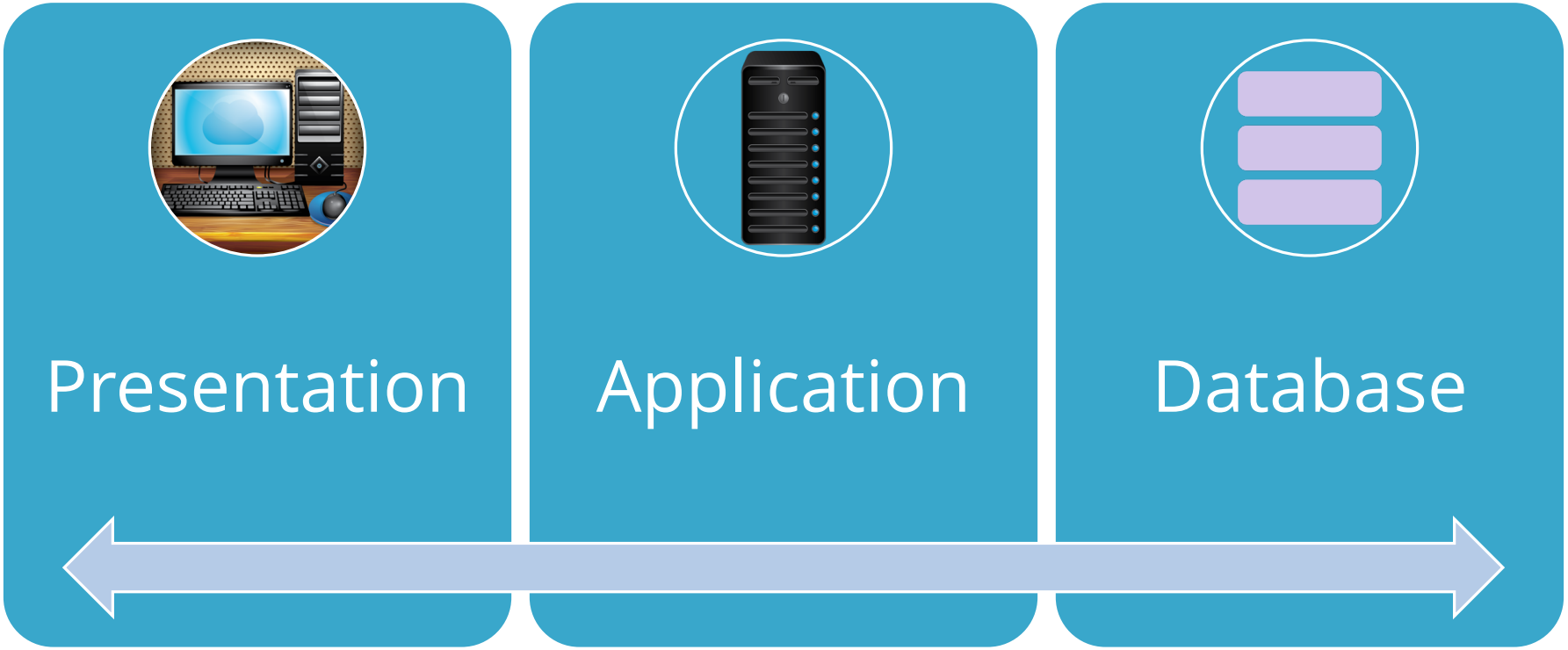
- Firebase
  - Yes, it's a turnkey BaaS provider
- Managed PostgreSQL
  - No, the developer is still responsible for scale
- AWS Lambda
  - Yes, it's a FaaS provider
- AWS DynamoDB
  - Yes, it fits into the BaaS camp, but does not provide all of the APIs that a turnkey BaaS provider has
- AWS Elastic Beanstalk
  - No, it is a PaaS that abstracts the use of underlying IaaS services, but the developer is still responsible for scaling concerns


Serverless Academy
Serverless Concepts
Traditional vs. Serverless Architectures

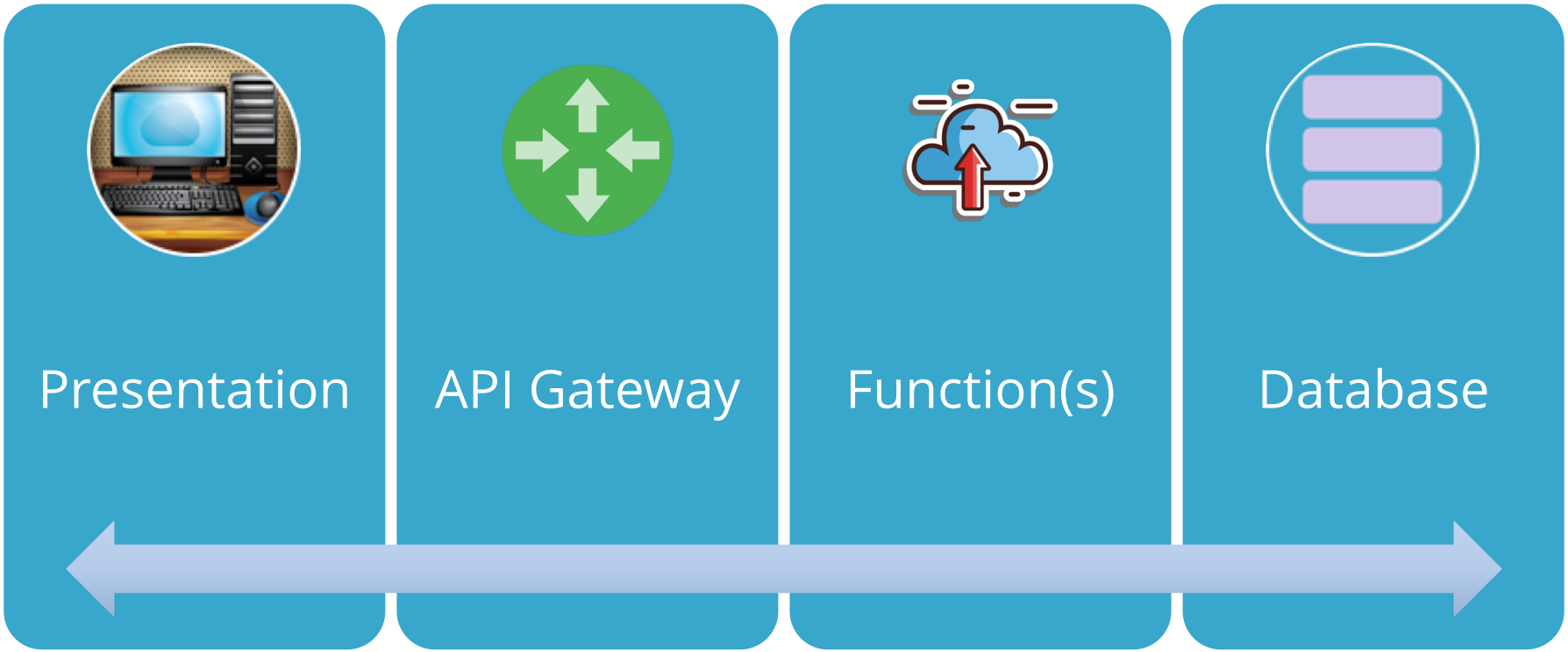
Eras of Provisioning Compute Units



Traditional Client/Server 3-Tier Architecture



Serverless Architecture



Developer Challenges

- Tooling is still very rough around the edges
  - Unless the dev team has strong opinions:
    - Function sprawl can impact
      - Developer workflow
      - Cognitive load
  - Infrastructure as Code best practices still forming

- Writing stateless code can be tricky
  - State is offloaded to a cache or database
  - If function relies on a large dataset, this can be a real challenge


Serverless Academy
Serverless Concepts
Serverless Offerings:
Google Firebase

Components Used in This BaaS

- Authentication
- Database
- Storage
- Hosting


Serverless Academy
Serverless Concepts
Serverless Offerings:
AWS Lambda

Possible Integrations

- Triggers:
  - API Gateway
  - AWS IoT
  - Alexa
  - CloudWatch
  - CodeCommit
  - Cognito
  - DynamoDB
  - Kinesis
  - S3
  - SNS

- All triggers to Lambda come from another AWS entry point

- This could cause concerns about vendor lock-in


Serverless Academy
Serverless Concepts
Serverless Offerings:
Azure Functions

Possible Integrations

- HTTP + Webhook
- Event processing
- Timer-based processing
- Stream Analytics


Serverless Academy
Serverless Concepts
Serverless Offerings:
IBM Bluemix OpenWhisk


Serverless Academy
Serverless Concepts
Serverless Offerings:
AWS API Gateway

What is an API Gateway?

- Thin layer of infrastructure that handles cross-cutting concerns:
  - Key Management
  - Route Definition
  - Request Throttling
  - Basic Caching

- API Gateway is considered serverless because it does not require the developer to be concerned with scale and maintenance


Serverless Academy
Serverless Concepts
Serverless Offerings:
On-Premise Options

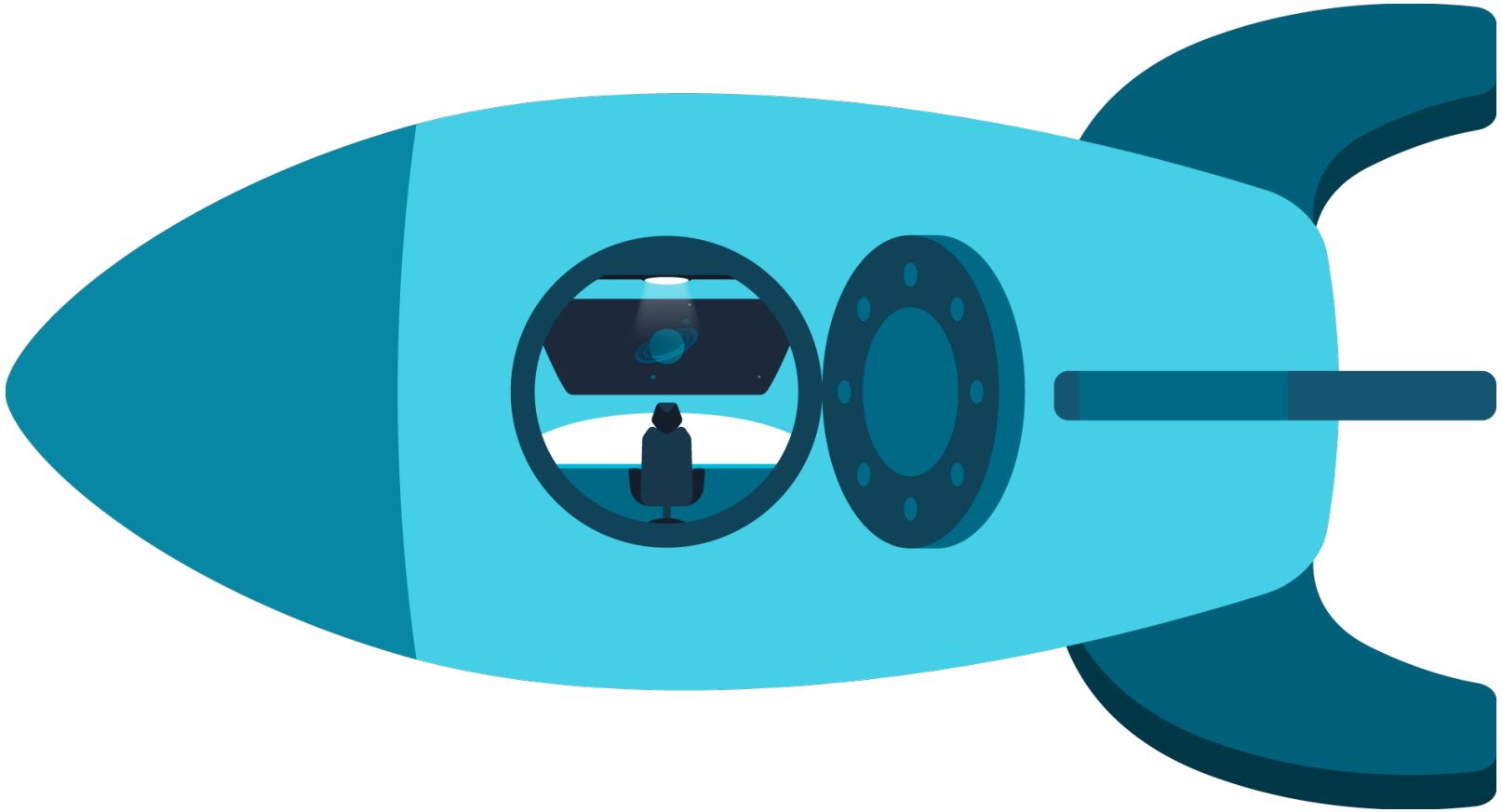
On-Premise Options

- Backend as a Service
  - Parse
- Function as a Service
  - OpenWhisk
  - Gesalt Framework


Parse

OpenWhisk

Gesalt Framework




Serverless Academy
Serverless Concepts
Benefits vs. Caveats

Benefits vs. Caveats

- Comparing Benefits and Caveats for:
  - Business
  - Developer
  - User

The Business Perspective: Benefits

- Cost is based on number of function executions, measured in milliseconds instead of hours
- Introduces more agility: Smaller deployable units results in faster delivery of features to the marketplace and increased ability to adapt to change
- Cost of hiring backend infrastructure engineers goes down

The Business Perspective: Caveats

- Less overall control:
  - Vendor lock-in requires more trust for a third-party provider
  - Additional exposure to risk requires more trust for a third-party provider
    - Security risk
    - Disaster recovery risk
  - Cost is unpredictable because number of executions are not predefined

- All of these caveats can be mitigated with open-source alternatives but at the expensive of the cost benefits mentioned previously

The Developer Perspective: Benefits

- No backend infrastructure to be responsible for
- Zero system administration
- Fosters adoption of:
  - Nanoservices
  - Microservices
  - SOA principles
- Faster to get up and running
- No need to worry about number of concurrent requests
- Get some monitoring out of the box

The Developer Perspective: Caveats

- Immature technology results in:
  - Component fragmentation
  - Unclear best-practices
- Discipline required against function sprawl
- Multitenancy means it’s technically possible that neighbor functions could hog the system resources behind the scenes
- Testing locally becomes tricky
- Further limitations:
  - Significant restrictions around local state
  - Execution duration is capped

The User Perspective: Benefits

- If businesses are using that competitive edge to ship features faster, then customers are receiving new features quicker than before

- If the front-end app becomes a data-shipping vehicle:
  - It's possible that users can more easily provide their own storage backend (ie: Dropbox)
  - It's more likely that these kind of apps may offer client-side caching, which provides a better offline experience

The User Perspective: Caveats

- Unless architected correctly, an app could provide a poor user experience as a result of increased request latency

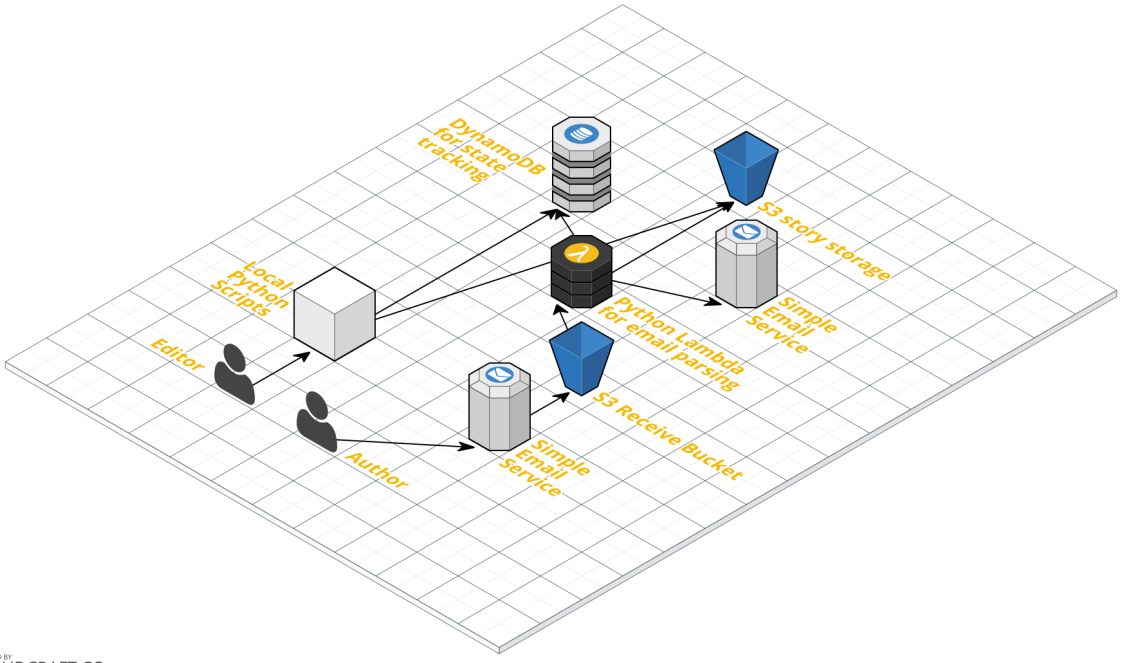

Serverless Academy
Serverless Concepts
Production Examples

AWS Serverless Application Model (SAM)

- Launched in November 2016
- Extends AWS CloudFormation to define resources for:
  - API Gateway
  - Lambda
  - DynamoDB

Experiences in Production

Experiences in Production



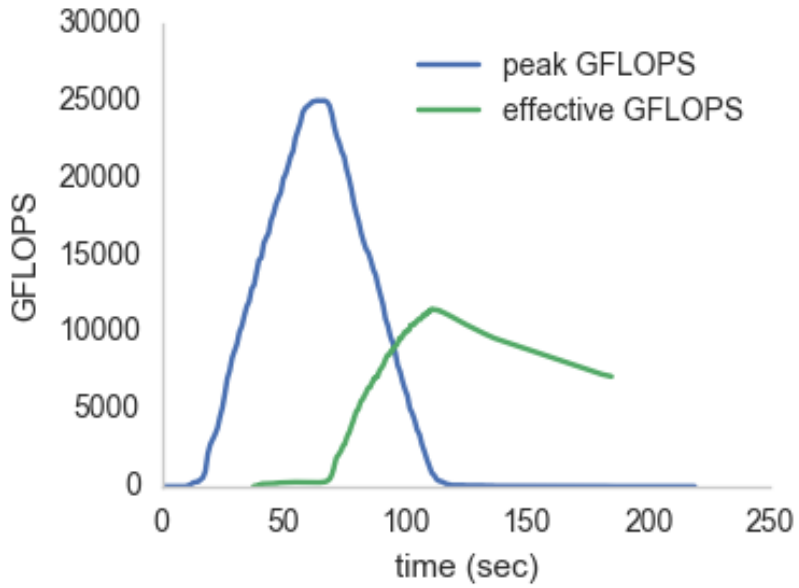
Compelling Science: Lessons Learned

- Run code without managing or provisioning servers
- Charges for compute time that is consumed
- Runs code in discrete chunks
- Takes care of everything required to run/scale with high availability
- Automatically trigger from other AWS services or from web/mobile

- More info: http://compellingsciencefiction.com/blog/2016-11-10.html

Experiences in Production

amplab
UC BERKELEY



AMP Lab: Lessons Learned

- Rate of submitting jobs is slow
- Most jobs start quickly after submission, but variance increases with number of running jobs
- Some jobs finish incredibly quickly, suggested they're running on faster/less-contested hardware

- More info:
  - http://ericjonas.com/pywren.html
  - http://tothestars.io/blog/2016/11/2/serverless-mapreduce


Serverless Academy
Serverless Concepts
Ways To Get Started

Adoption Paths

- Adoption will look different based on some varying factors:
  - Greenfield or legacy
  - Simple or complex
  - Compliance requirements

- Greenfield projects can pick up a framework and get started
- Legacy projects need to be a little more strategic
  - Evaluate dependencies
  - Apply the Strangler Pattern to your application

Frameworks

- Serverless Framework
- Apex
- ClaudiaJS
- Sparta
- Gordon
- Zappa

Frameworks: Serverless

Frameworks: APEX

Frameworks: Claudia.js

Frameworks: Sparta

Frameworks: Gordon

Frameworks: Zappa