TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing Concepts and Models

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

OBJECTIVES

- Term project proposal
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models

OBJECTIVES

- Term project proposal
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models

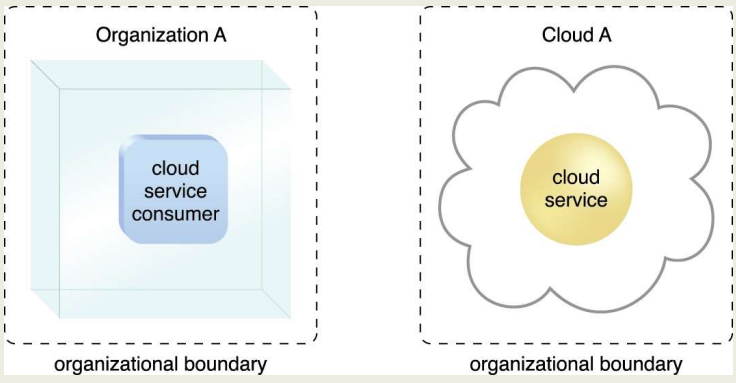
ROLES

- **Cloud provider**
  - Organization that provides cloud-based resources
  - Responsible for fulfilling SLAs for cloud services
  - Some cloud providers "resell" IT resources from other cloud providers
    - Example: Heroku sells PaaS services running atop of Amazon EC2
- **Cloud consumers**
  - Cloud users that consume cloud services
- **Cloud service owner**
  - Both cloud providers and cloud consumers can own cloud services
  - A cloud service owner may use a cloud provider to provide a cloud service (e.g. Heroku)

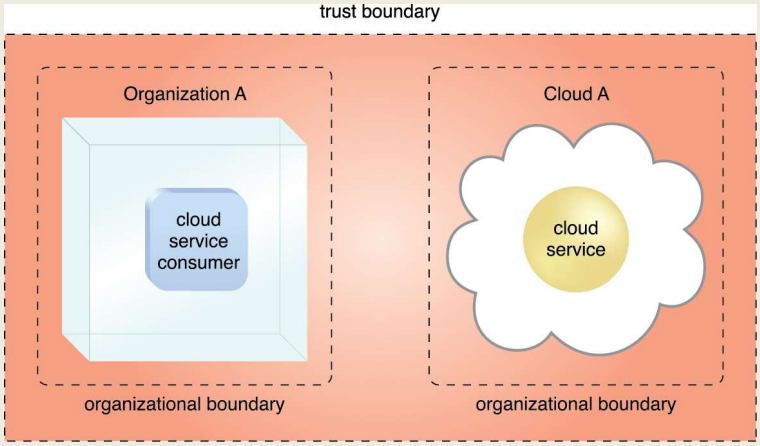
ROLES - 2

- **Cloud resource administrator**
  - Administrators provide and maintain cloud services
  - Both cloud providers and cloud consumers have administrators
- **Cloud auditor**
  - Third-party which conducts independent assessments of cloud environments to ensure security, privacy, and performance.
  - Provides unbiased assessments
- **Cloud brokers**
  - An intermediary between cloud consumers and cloud providers
  - Provides service aggregation
- **Cloud carriers**
  - Network and telecommunication providers which provide network connectivity between cloud consumers and providers

ORGANIZATION BOUNDARY



TRUST BOUNDARY



OBJECTIVES

- Term project proposal
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - **Cloud characteristics**
  - Cloud delivery models
  - Cloud deployment models

CLOUD CHARACTERISTICS

- On-demand usage
- Ubiquitous access
- Multitenancy (resource pooling)
- Elasticity
- Measured usage
- Resiliency

- Assessing these features helps measure the value offered by a given cloud service or platform

ON-DEMAND USAGE

- The freedom to self-provision IT resources
- Generally with automated support
- Automated support requires no human involvement
- Automation through software services interface

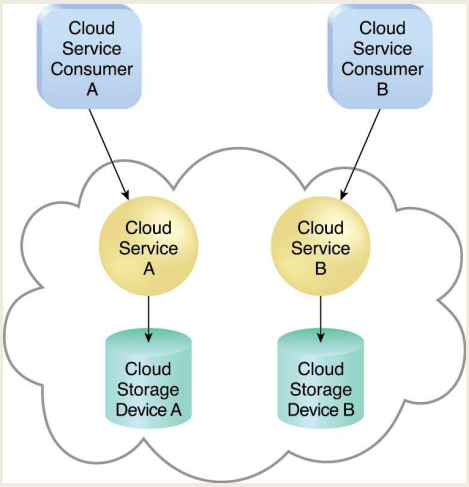
UBIQUITOUS ACCESS

- Cloud services are widely accessible
- Public cloud: internet accessible
- Private cloud: throughout segments of a company's intranet
- 24/7 availability

MULTITENANCY

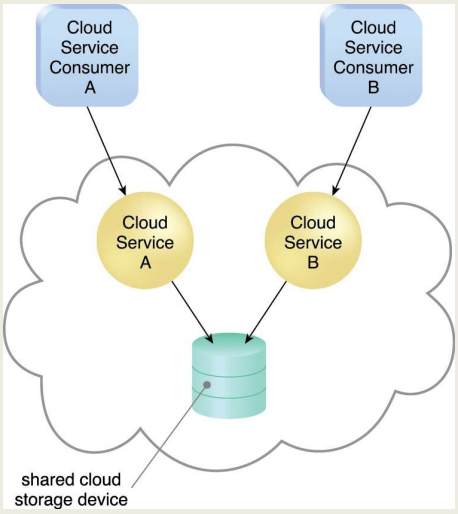
- Cloud providers pool resources together to share them with many users
- Serve multiple cloud service consumers
- IT resources can be dynamically assigned, reassigned based on demand
- Multitenancy can lead to performance variation

SINGLE TENANT MODEL

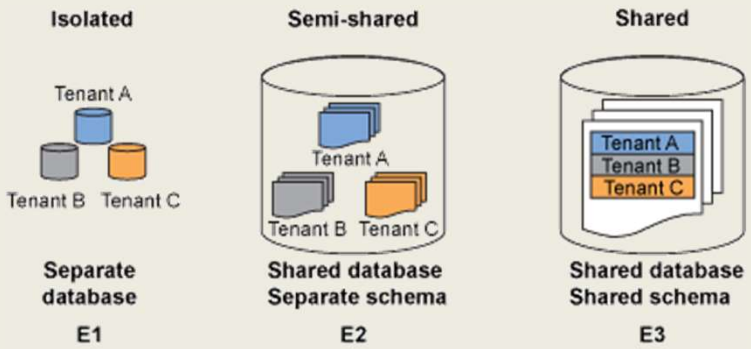


MULTITENANT MODEL

- Resource is "multiplexed" and share amongst multiple users
- Goal is to increase utilization
- Often server resources are underutilized
- There are many "sunk costs" whether usage is 0% or 100%
- Cloud computing tries to maximize "sunk cost" investments

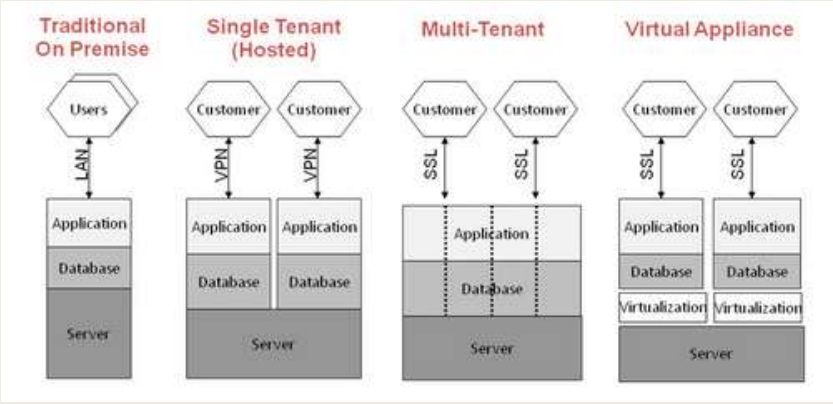


MULTITENANT DATABASE



MULTITENANCY OF RESOURCES

- Where is the multitenancy?

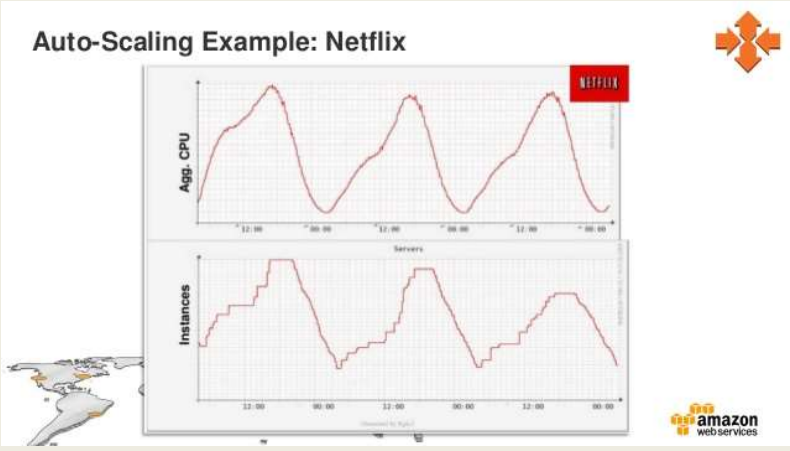


ELASTICITY

- Automated ability of cloud to transparently scale resources
- Scaling based on runtime conditions or pre-determined by cloud consumer or cloud provider
- Threshold based scaling
  - `CPU-utilization > threshold_A, Response_time > 100ms`
  - Application agnostic vs. application specific thresholds
  - Why might an application agnostic threshold be non-ideal?
- Load prediction
  - Historical models
  - Real-time trends

PREDICTABLE DEMAND

- Example:

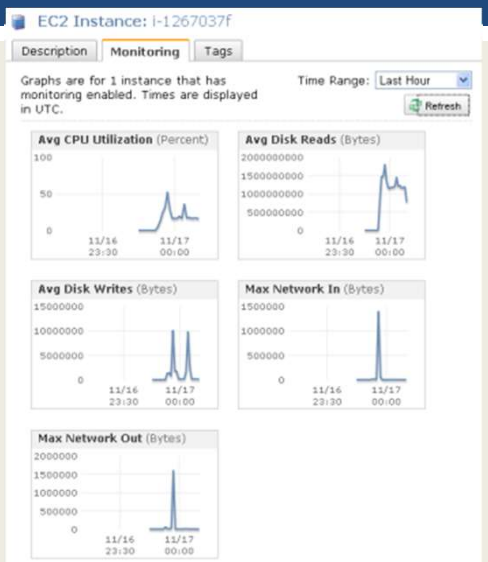


MEASURED USAGE

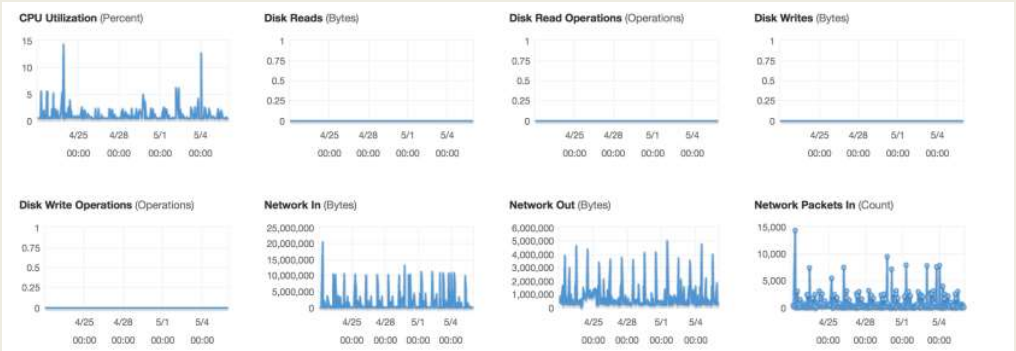
- Cloud platform tracks usage of IT resources
- For billing purposes
- Enables charging only for IT resources actually used
- Can be time-based (minute, hour, day)
- Can be throughput-based (MB, GB)

- Not all measurements are for billing
- Some measurements can support auto-scaling
- For example CPU utilization

EC2 CLOUDWATCH METRICS

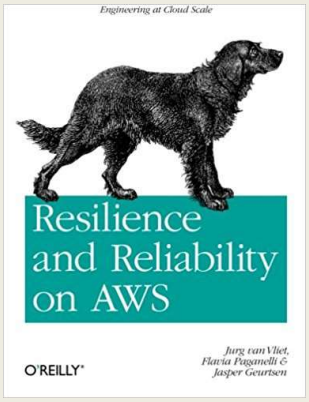


EC2 CLOUDWATCH METRICS



RESILIENCY

- Distributed redundancy across physical locations
- Used to improve reliability and availability of cloud-hosted applications
- Very much an engineering problem
- No "resiliency-as-a-service" for user deployed apps
- Unique characteristics of user applications make a one-size fits all service solution challenging

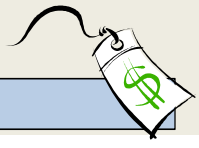


OBJECTIVES

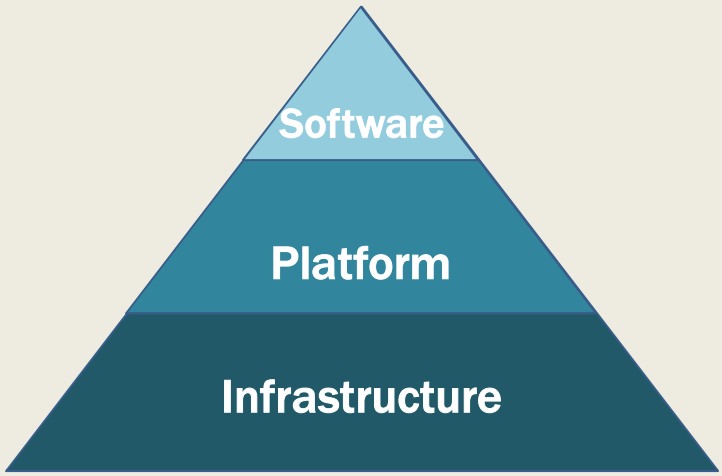
- Term project proposal
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - **Cloud delivery models**
  - Cloud deployment models

CLOUD DELIVERY MODELS

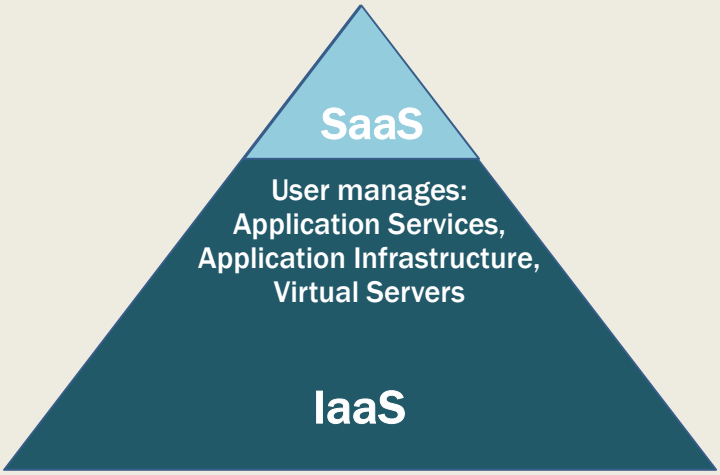
- What is the appropriate level of **abstraction**?
- How should applications be deployed?
  - IaaS, PaaS, SaaS, DbaaS, FaaS
- How do we ensure Quality-of-Service?
  - Performance, Availability, Responsiveness, Fault Tolerance
- How is ***scalability*** provided?
- How do we minimize hosting costs?
  - How do we estimate hosting costs?



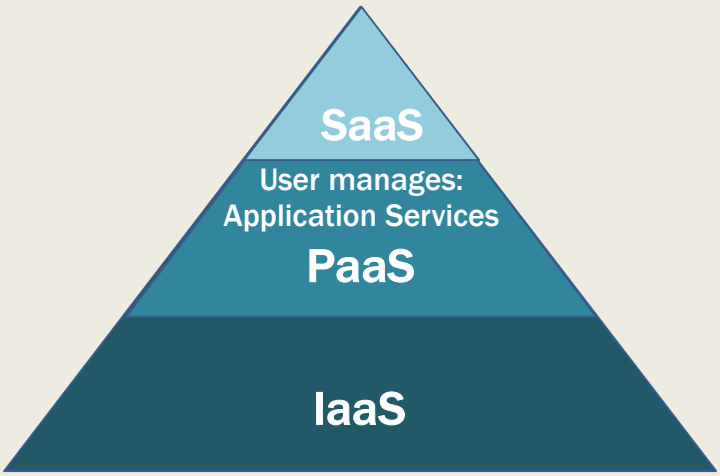
CLOUD DELIVERY MODELS



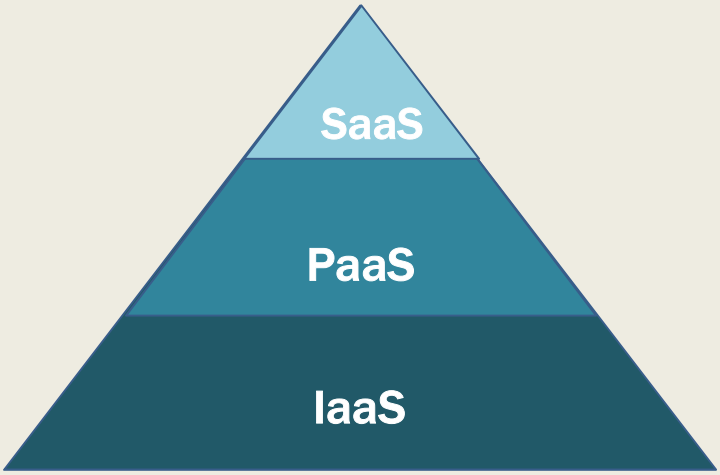
CLOUD DELIVERY MODELS



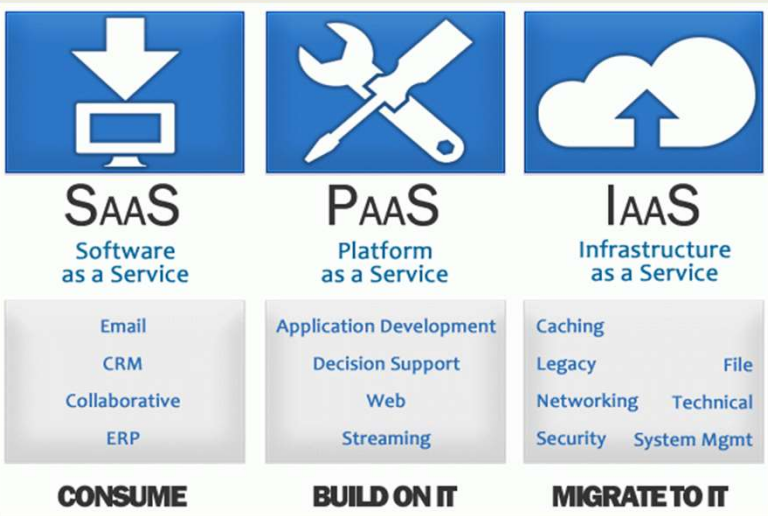
CLOUD DELIVERY MODELS

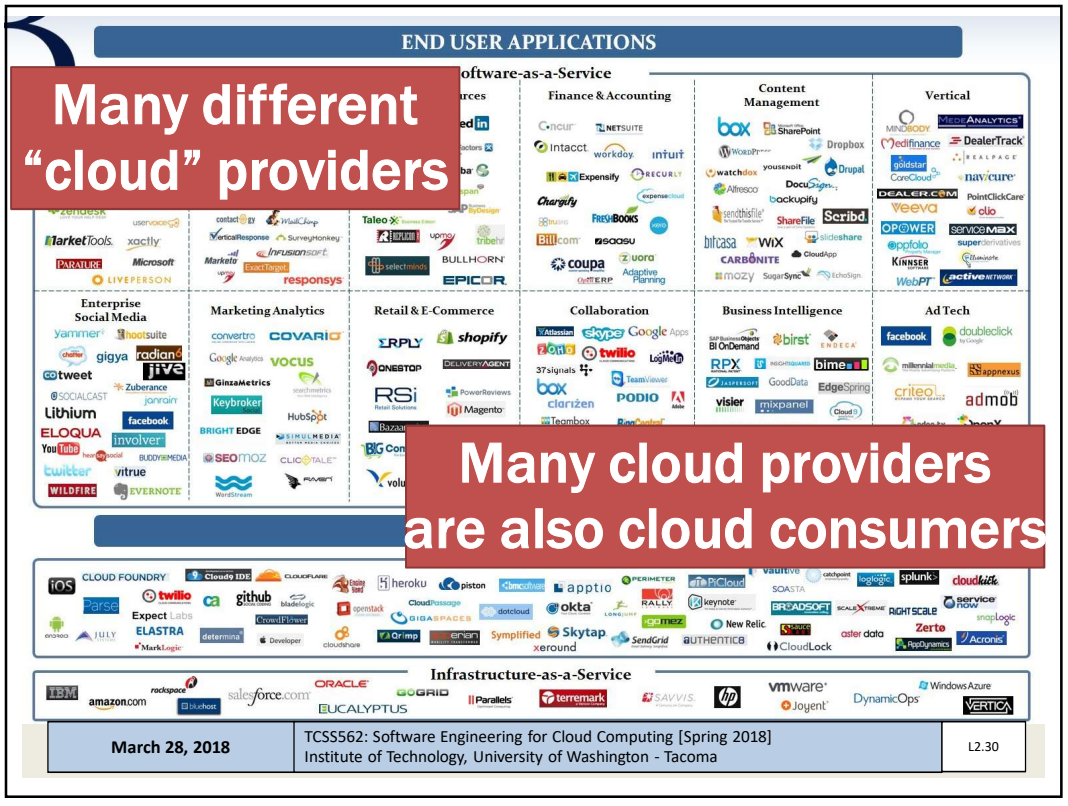


CLOUD DELIVERY MODELS



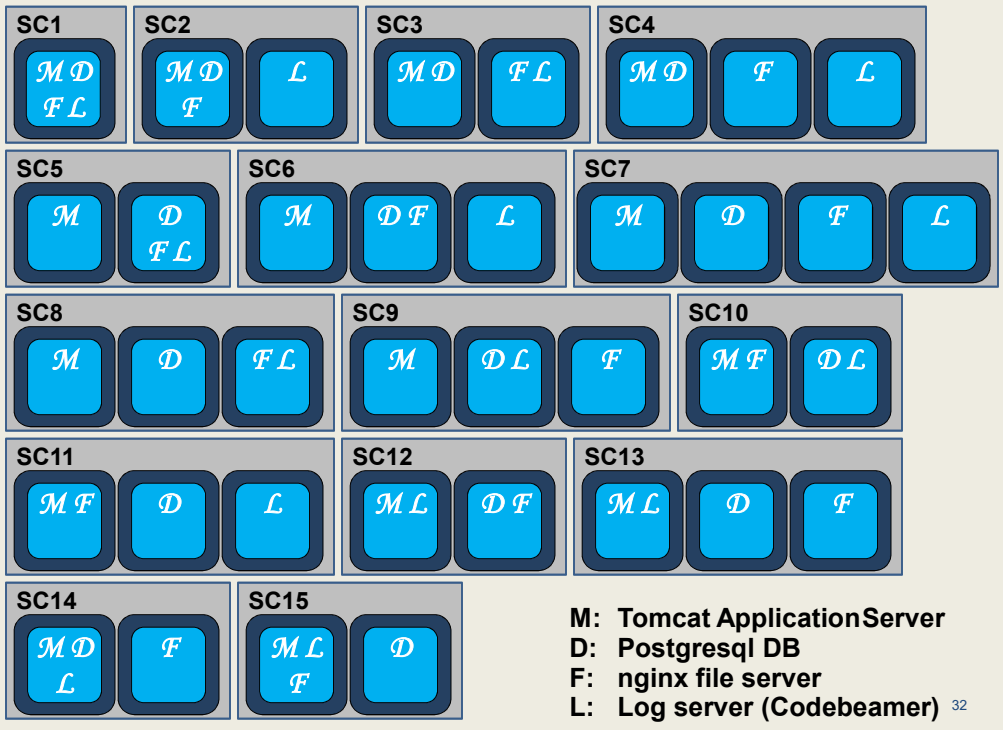
EXAMPLE CLOUD SERVICES

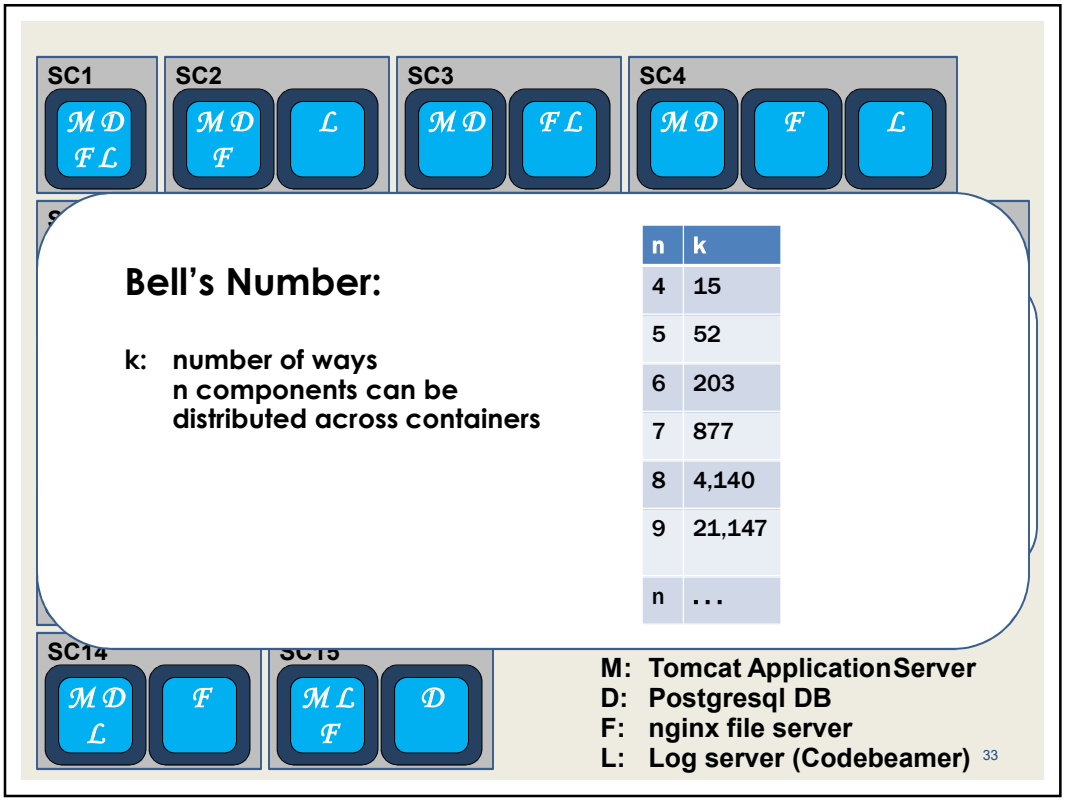


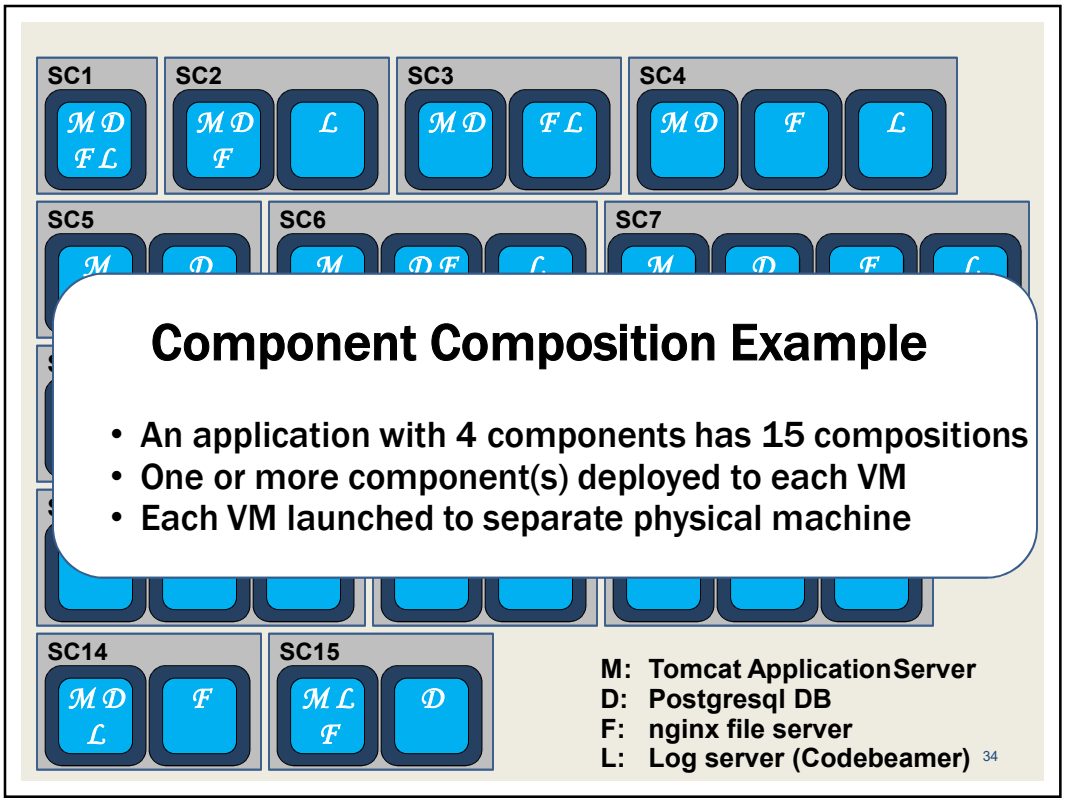


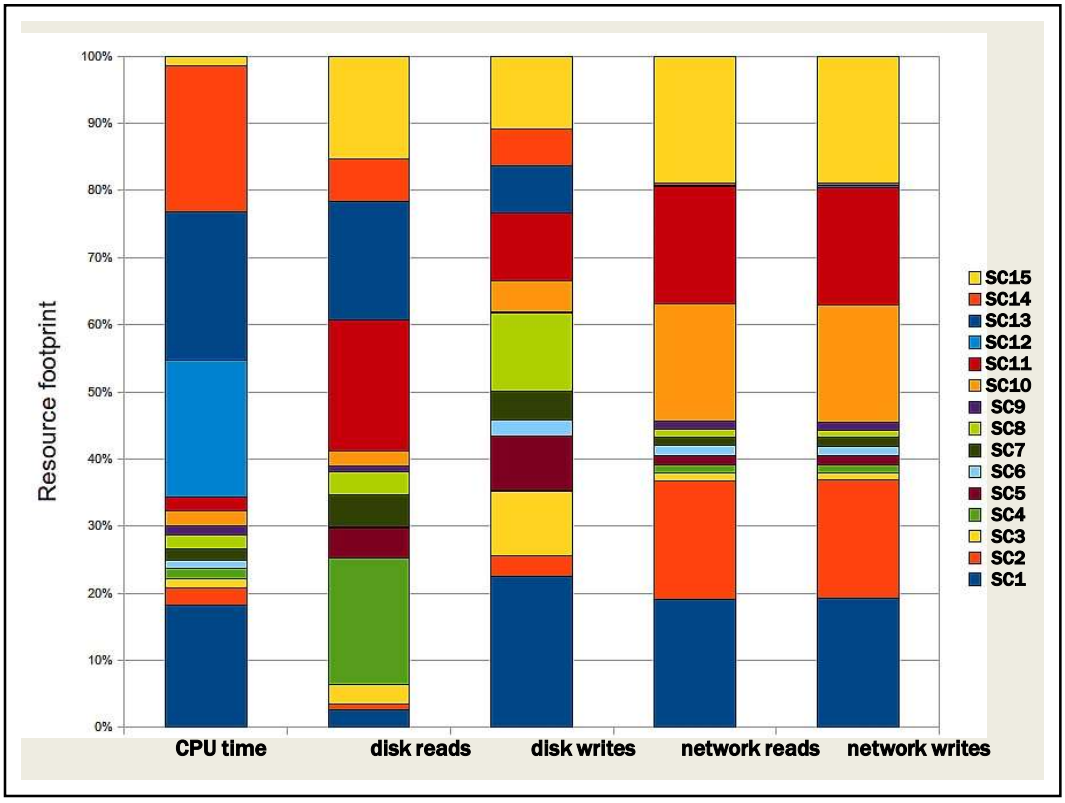
INFRASTRUCTURE-AS-A-SERVICE

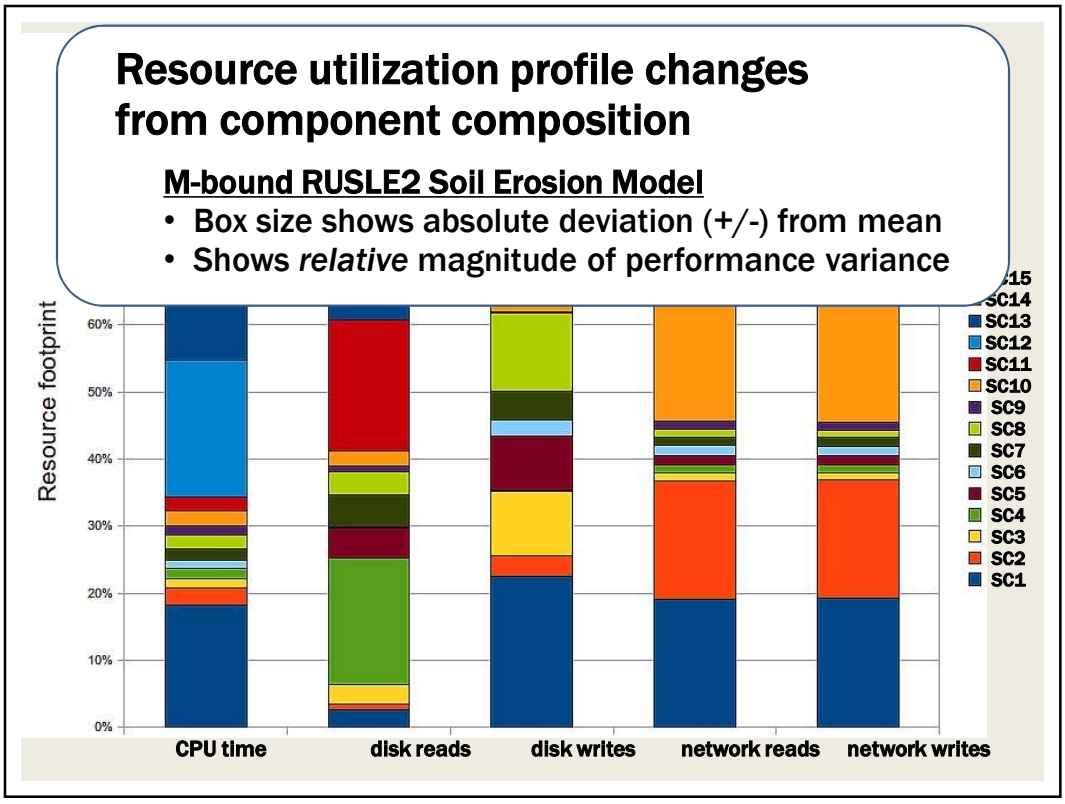
- Compute resources, on demand, as-a-service
  - Generally raw "IT" resources
  - Hardware, network, containers, operating systems
- Typically provided through virtualization
- Generally not-preconfigured
- Administrative burden is owned by cloud consumer
- Best when high-level control over environment is needed
- Scaling is generally **not** automatic…
- Resources can be managed in bundles
- AWS CloudFormation: Allows specification in JSON/YAML of cloud infrastructures

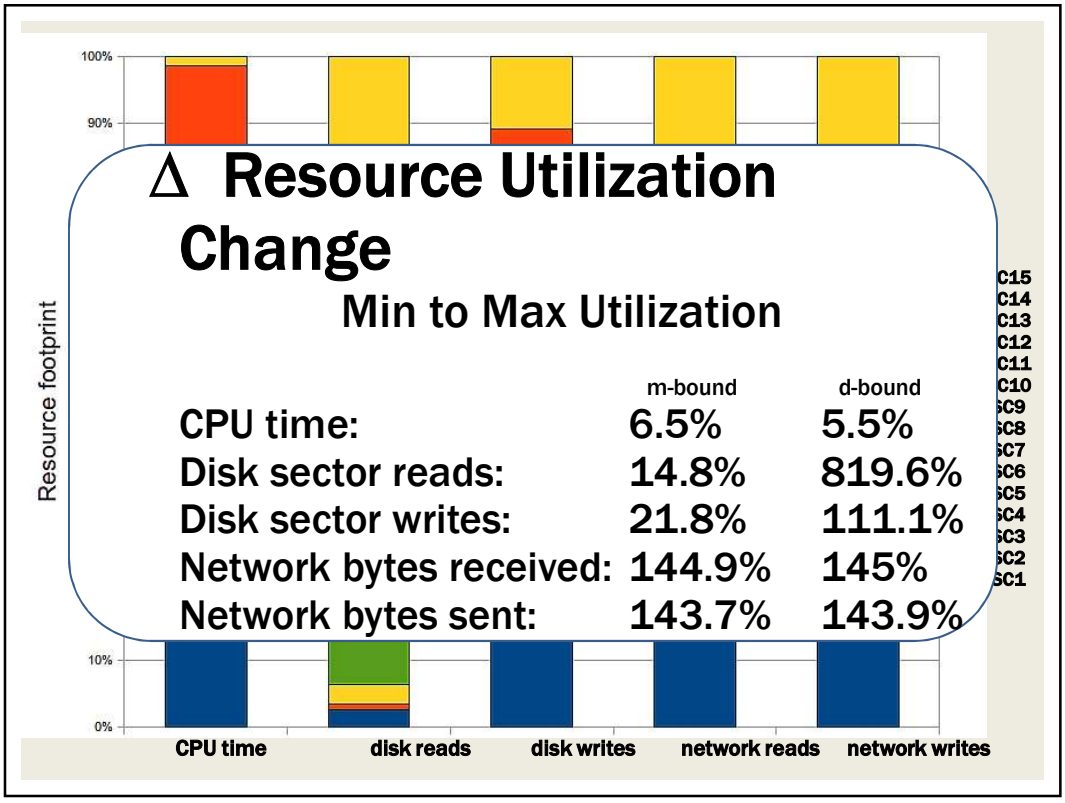



SC1
SC2
SC3
SC4
SC14
SC15
Bell's Number:
k: number of ways
n components can be
distributed across containers
n k
4 15
5 52
6 203
7 877
8 4,140
9 21,147
n ...
M: Tomcat ApplicationServer
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)
33


SC1
SC2
SC3
SC4
SC5
SC6
SC7
SC14
SC15
Component Composition Example
• An application with 4 components has 15 compositions
• One or more component(s) deployed to each VM
• Each VM launched to separate physical machine
M: Tomcat ApplicationServer
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)
34


Resource footprint
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
CPU time
disk reads
disk writes
network reads
network writes
SC15
SC14
SC13
SC12
SC11
SC10
SC9
SC8
SC7
SC6
SC5
SC4
SC3
SC2
SC1

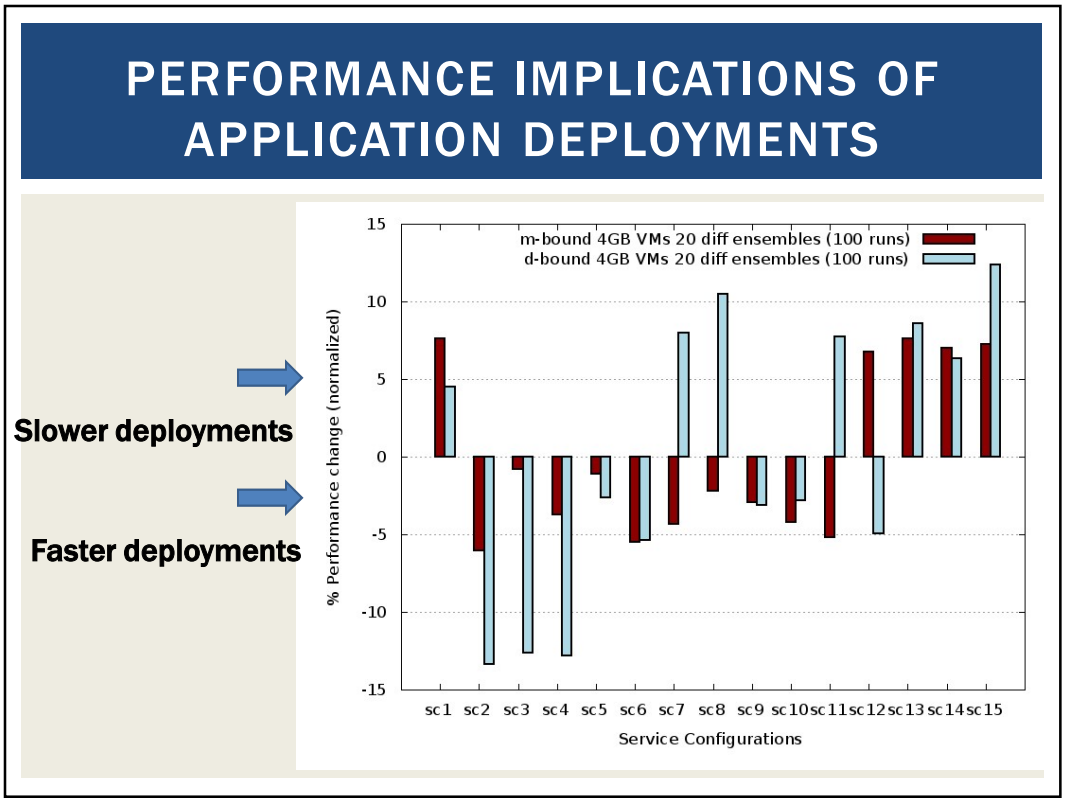

Resource utilization profile changes from component composition
M-bound RUSLE2 Soil Erosion Model
• Box size shows absolute deviation (+/-) from mean
• Shows *relative* magnitude of performance variance
Resource footprint
60%
50%
40%
30%
20%
10%
0%
CPU time
disk reads
disk writes
network reads
network writes
SC13
SC12
SC11
SC10
SC9
SC8
SC7
SC6
SC5
SC4
SC3
SC2
SC1



Δ Resource Utilization Change

Min to Max Utilization

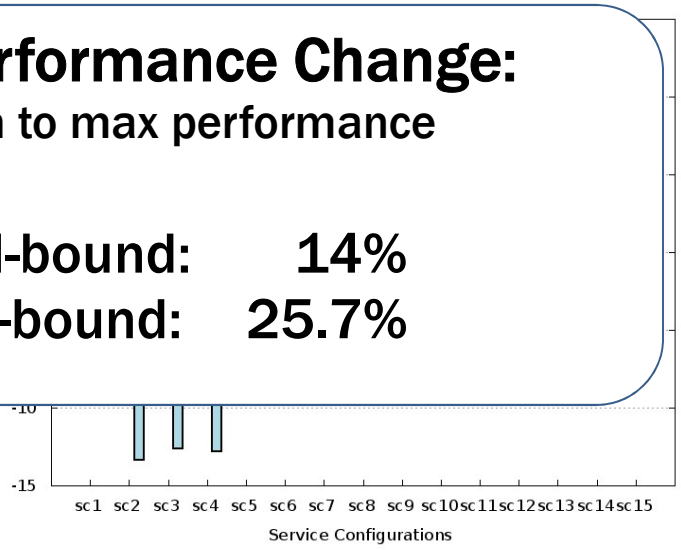
	m-bound	d-bound
CPU time:	6.5%	5.5%
Disk sector reads:	14.8%	819.6%
Disk sector writes:	21.8%	111.1%
Network bytes received:	144.9%	145%
Network bytes sent:	143.7%	143.9%



PERFORMANCE IMPLICATIONS OF APPLICATION DEPLOYMENTS

Δ Performance Change:
Min to max performance

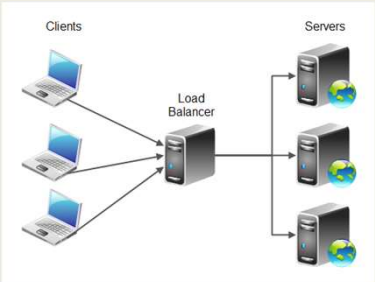
M-bound: 14%
D-bound: 25.7%



PLATFORM-AS-A-SERVICE

- Predefined, ready-to-use, hosting environment
- Infrastructure is further obscured from end user
- Scaling and load balancing may be automatically provided and automatic
- Variable to no ability to influence responsiveness

- Examples:
- Google App Engine
- Heroku
- AWS Elastic Beanstalk
- AWS Lambda (FaaS)

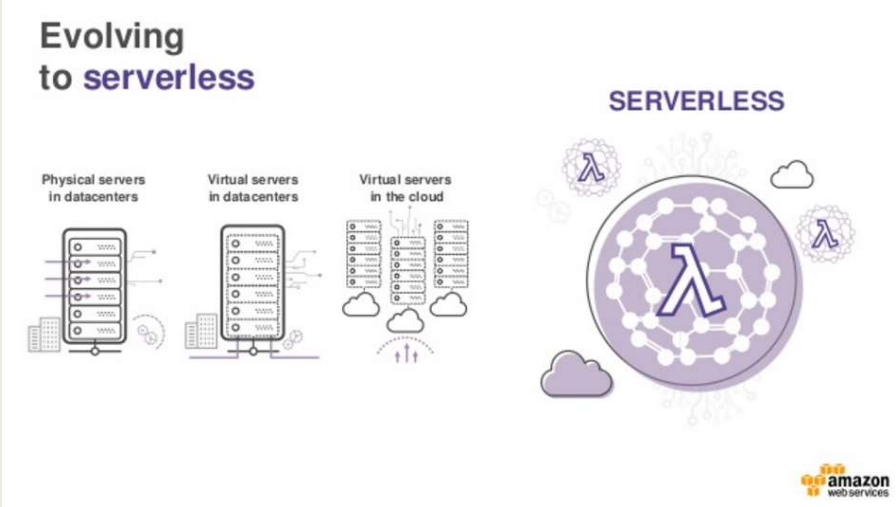


USES FOR PAAS

- Cloud consumer
  - Wants to extend on-premise environments into the cloud for "web app" hosting
  - Wants to entirely substitute an on-premise hosting environment
  - Cloud consumer wants to become a cloud provider and deploy its own cloud services to external users
- PaaS spares IT administrative burden compared to IaaS

SERVERLESS COMPUTING

SERVERLESS COMPUTING - 2



SERVERLESS COMPUTING - 3

- New cloud platform for hosting application code
- Every cloud vendor provides their own:
  - AWS Lambda, Azure Functions, Google Cloud Functions, IBM OpenWhisk
- Similar to platform-as-a-service
- Replace opensource web container (e.g. Apache Tomcat) with abstracted vendor-provided **black-box** environment

SERVERLESS COMPUTING - 4

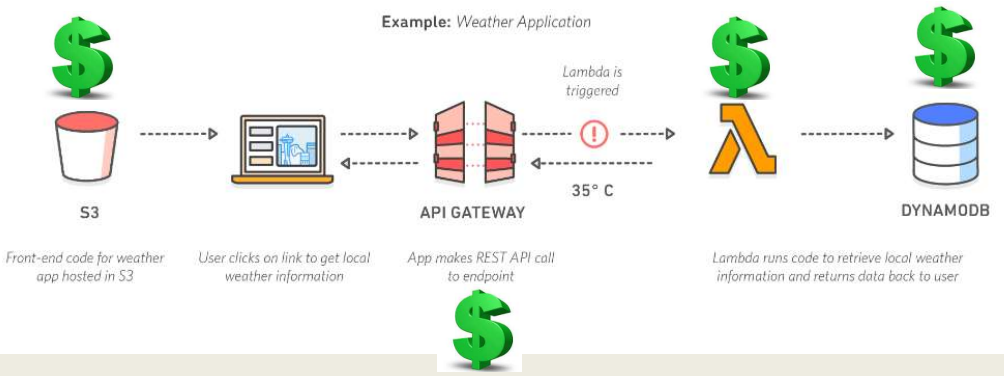
- Many challenging features of distributed systems are provided automatically
- ***Built into the platform:***
- Highly availability (24/7)
- Scalability
- Fault tolerance

SERVERLESS COMPUTING

AWS Lambda Demo

SERVERLESS SOFTWARE ARCHITECTURE

- Every service with a different pricing model



IAAS BILLING MODELS

- Virtual machines as-a-service at ¢ per hour
- No premium to scale:

```
        1000 computers    @     1 hour
  =        1 computer     @  1000 hours
```

- Illusion of infinite scalability to cloud user
- As many computers as you can afford
- Billing models are becoming increasingly granular
  - By the minute, second, 1/10th sec
- Auction-based instances: Spot instances →

QUESTIONS

March 28, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L2.49