Amazon DynamoDB

Devdatta Kulkarni

What is DynamoDB

- Fully managed NoSQL database service

- What does Fully managed mean?
  - Managed by AWS – application developers don't have to create and manage the DB instance
- NoSQL database
  - Database that does not require strict schemas like traditional SQL databases (RDBMS)
    - MySQL, Postgres, etc.

- https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

High-level feature summary

- Define table throughput capacity for each table
- On demand backup
- Encryption at rest
- Delete expired items from table automatically
- Data is automatically replicated across multiple Availability Zones in an AWS region
- Global tables feature that allows to keep DynamoDB tables in sync across AWS regions

Core Concepts

- Table
- Items
- Attributes
- A table is a collection of items
- Each item is a collection of attributes

- Item == row
- Attribute == Column
- Each item has a Primary key attribute which uniquely identifies each item in a table
- The primary key attribute needs to be present in each item in the table
- Other attributes in each item need not be same

Naming Rules

- All names are case-sensitive
- Table and Index names must be between 3 - 255 characters long
- Attribute names must be between 1 - 255 characters long

Data types

- An attribute can have one of the following kinds of data
  - Scalar types
    - String, Number, Binary, Boolean, Null
  - Document types
    - List and Map
  - Set types
    - Multiple scalar values
- When creating table or index, the data type of each primary key attribute needs to be specified
- A primary key attribute can be only of following types:
  - String, Number, Binary

People Table

```
{
    “PersonID”: 101,
    “LastName”: “Smith”,
    “FirstName”: “Fred”,
    “Phone”: “555-4321”
}
```

```
{
    “PersonID”: 102,
    “LastName”: “Jones”,
    “FirstName”: “Mary”,
    “Address”: {
         “Street”: “123 Lamar”,
         “City”: “Austin”,
         “State”: “TX”,
         “Zipcode”: “78737”
     }
}
```

PersonID: Primary key Attribute

Each Item needs to have the Primary key Attribute

Different Items may have different Attributes

Nested Attributes are possible – DynamoDB supports 32 level deep nesting

Composite Primary key (consisting of multiple Attributes) is possible

Primary Key

- Primary Key attribute needs to be specified when creating a Table

- Two different kinds of Primary Keys
  - Partition Key
    - Composed of one attribute
    - Used with a hash function to determine where to store the item
    - Only one item can have a specific value for the primary key
  - Partition Key and sort Key
    - Composite primary key (composed of two attributes)
    - First attribute is the Partition key (also called 'hash attribute')
    - Second attribute is the Sort key (also called 'range attribute')
    - All items with the same partition key are stored together, in sorted order by sort key value
    - Multiple items can have same value for primary key, but they should have different value for the sort key

Primary Key

- A Primary Key attribute must be such that it holds a single value
  - String
  - Number
  - Binary
- Primary Key attribute(s) are typically used for Querying data

Primary Key

{
    “**PersonID**”: 101,
    “LastName”: “Smith”,
    “FirstName”: “Fred”,
    “Phone”: “555-4321”
}

{
    “**PersonID**”: 101,
    “LastName”: “Jones”,
    “FirstName”: “Mary”,
    “Phone”: “555-4321”
}

Incorrect: Only one item can have a specific value for **PersonID** attribute

{
    “**LastName**”: “Smith”,
    “**FirstName**”: “Fred”,
    “Phone”: “555-4321”
}

{
    “**LastName**”: “Smith”,
    “**FirstName**”: “Mary”,
    “Phone”: “555-4321”
}

Correct: **LastName** is same, **FirstName** is different

<LastName, FirstName>: Composite Primary Key
LastName: Partition Key
FirstName: Sort Key

API

- Control Plane
  - To work with tables and streams
- Data Plane
  - To work with data items in the table
- Streams
  - To work with streams

Control Plane API

- CreateTable
- DescribeTable
- ListTables
- UpdateTable
- DeleteTable

CreateTable

```
table = dynamodb.create_table(
    TableName='Movies',
    KeySchema=[
        {
            'AttributeName': 'year',
            'KeyType': 'HASH'  #Partition key
        },
        {
            'AttributeName': 'title',
            'KeyType': 'RANGE'  #Sort key
        }
    ],
    AttributeDefinitions=[
        {
            'AttributeName': 'year',
            'AttributeType': 'N'
        },
        {
            'AttributeName': 'title',
            'AttributeType': 'S'
        },

    ],
    ProvisionedThroughput={
        'ReadCapacityUnits': 10,
        'WriteCapacityUnits': 10
    }
)

print("Table status:", table.table_status)
```

Data Plane API

- PutItem
- BatchWriteItem
- GetItem
- BatchGetItem
- Query
  - Partition key value needs to be provided; Optionally takes sort key as input
- Scan
  - Retrieve all items in the specified table or index
- UpdateItem
- DeleteItem
- BatchWriteItem

PutItem

```
table = dynamodb.Table('Movies')

title = "The Big New Movie"
year = 2015

response = table.put_item(
   Item={
        'year': year,
        'title': title,
        'info': {
            'plot':"Nothing happens at all.",
            'rating': decimal.Decimal(0)
        }
    }
)
```

Primary key (year, title) is required

GetItem

```
table = dynamodb.Table('Movies')

title = "The Big New Movie"
year = 2015

try:
    response = table.get_item(
        Key={
            'year': year,
            'title': title
        }
    )
except ClientError as e:
    print(e.response['Error']['Message'])
else:
    item = response['Item']
    print("GetItem succeeded:")
    print(json.dumps(item, indent=4, cls=DecimalEncoder))
```

Get Item - Primary key (year, title) is required

Query

```
table = dynamodb.Table('Movies')

print("Movies from 1985")

response = table.query(
    KeyConditionExpression=Key('year').eq(1985)
)

for i in response['Items']:
    print(i['year'], ":", i['title'])
```

Hash key (year) needs to be specified in the Query call

KeyConditionExpression → Type of ConditionExpression

Query

```
table = dynamodb.Table('Movies')

print("Movies from 1992 - titles A-L, with genres and lead actor")

response = table.query(
    ProjectionExpression="#yr, title, info.genres, info.actors[0]",
    ExpressionAttributeNames={ "#yr": "year" }, # Expression Attribute Names for Projection
Expression only.
    KeyConditionExpression=Key('year').eq(1992) & Key('title').between('A', 'L')
)

for i in response[u'Items']:
    print(json.dumps(i, cls=DecimalEncoder))
```

- Query needs Partition key to be specified (done through KeyConditionExpression)

- ProjectionExpression → Indicates the list of attributes that need to be output in the result

- ExpressionAttributeNames → Used for Name substitution – In DynamoDB 'year' is a reserved name, so we cannot use it in ProjectionExpression, FilterExpression, ConditionExpression, UpdateExpression

Condition Expressions

- A condition expression is used to determine which items should be modified (PutItem, DeleteItem, etc.)
- If the condition expression evaluates to true, the operation succeeds; otherwise, the operation fails

Condition Expression: Examples

```
aws dynamodb delete-item \
    --table-name ProductCatalog \
    --key '{"Id": {"N": "456"}}' \
    --condition-expression "attribute_not_exists(Price)"
```

```
aws dynamodb delete-item \
    --table-name ProductCatalog \
    --key '{"Id": {"N": "456"}}' \
    --condition-expression "attribute_exists(ProductReviews.OneStar)"
```

Condition Expressions: Example

```
aws dynamodb delete-item \
    --table-name ProductCatalog \
    --key '{"Id":{"N":"456"}}' \
    --condition-expression "(ProductCategory IN (:cat1, :cat2)) and (Price between :lo and :hi)" \
    --expression-attribute-values file://values.json
```

```
{
    ":cat1": {"S": "Sporting Goods"},
    ":cat2": {"S": "Gardening Supplies"},
    ":lo": {"N": "500"},
    ":hi": {"N": "600"}
}
```

: → Indicates an Expression Attribute Value. It acts as a placeholder for the actual value that will be provided later

Use Expression Attribute Values if you don't know the actual value to use in a comparison till runtime

Conditional Updates

```
aws dynamodb update-item \
    --table-name ProductCatalog \
    --key '{"Id": {"N": "456"}}' \
    --update-expression "SET Price = Price - :discount" \
    --condition-expression "Price > :limit" \
    --expression-attribute-values file://values.json
```

```
{
    ":discount": { "N": "75"},
    ":limit": {"N": "500"}
}
```

Reduce the Price of a product by 75 if the current Price is above 500

Expression Attribute Values

- Expression attribute values are substitutes for the actual values that you want to compare — values that you might not know until runtime.

- An expression attribute value must begin with a :

- Expression attribute values are used with condition expressions, update expressions, and filter expressions

Expression Attribute Values

```
aws dynamodb scan \
    --table-name ProductCatalog \
    --filter-expression "contains(Color, :c) and Price <= :p" \
    --expression-attribute-values file://values.json
```

```
{
    ":c": { "S": "Black" },
    ":p": { "N": "500" }
}
```

Filter products from ProductCatalog that are Black in Color and have Price less than or equal to 500

Scan

```
table = dynamodb.Table('Movies')

fe = Key('year').between(1950, 1959);
pe = "#yr, title, info.rating"
# Expression Attribute Names for Projection Expression only.
ean = { "#yr": "year", }
esk = None


response = table.scan(
    FilterExpression=fe,
    ProjectionExpression=pe,
    ExpressionAttributeNames=ean
    )

for i in response['Items']:
    print(json.dumps(i, cls=DecimalEncoder))

while 'LastEvaluatedKey' in response:
    response = table.scan(
        ProjectionExpression=pe,
        FilterExpression=fe,
        ExpressionAttributeNames= ean,
        ExclusiveStartKey=response['LastEvaluatedKey']
        )

    for i in response['Items']:
        print(json.dumps(i, cls=DecimalEncoder))
```

- No Partition key needs to be specified

- Scan method reads every item in the entire table and returns all the Items

- FilterExpression can be used to limit the number of items that are returned

- FilterExpression is applied only after the table is scanned

- Each scan returns a 'Page' of keys

- LastEvaluatedKey should be passed to subsequent scans as the ExclusiveStartKey

Read Consistency

- Tables created in one region is completely separate from tables in other regions
  - You can have table with same name in multiple regions
- Table data is replicated in multiple availability zones within a region
- When data is written, all the replicas are updated eventually
  - Success reply is sent back to the client after at least W replicas have been updated
  - Inconsistency window: 1 second or less
- DynamoDB supports *eventually consistent* reads and *strongly consistent* reads

Read Consistency

- Controlled by “ConsistentRead” parameter available for:
  - GetItem, Query, and Scan operations
- Default value is False
  - This will give eventually consistent read behavior
  - This means it is possible that you will see stale data
- When set to True
  - We get strongly consistent read behavior
  - This means we will always get most up-to-date data
  - Strong consistent read is not available if there are network partitions

Throughput capacity for Reads / Writes

- Notion of throughput capacity
  - How many read/write operations would be done on the table/index per second
- Need to be specified per table/index at the time of creation
- One read capacity unit
  - 1 strongly consistent read/second for data item of up to 4 KB
  - 2 eventually consistent read/second for data item of up to 4 KB
- One write capacity unit
  - 1 write per second for data item of up to 1 KB

Throughput capacity

- DynamoDB maximum item size
  - 400 KB
  - Implication
    - We cannot use DynamoDB to store large data items such as images
    - S3 is the better and preferred storage for binary data
- Throttling
  - DynamoDB will throttle read/write requests if the request rate goes beyond the declared read/write capacity throughput

Throughput capacity management

- Auto scaling
  - Define a range (upper and lower limits) for read and write capacity units
  - Define target utilization percentage within the range
  - Request throttling is not used
    - Throughput is increased or decreased dynamically
- Provisioned throughput
  - Maximum amount of capacity that application can consume from a table/index
  - Request throttling is done if application exceeds the defined capacity
- Reserved capacity
  - Can be purchased to avoid request throttling

Example

- Table with 5 read capacity units and 5 write capacity units
  - How much data can be added per second?
    - 1 KB * 5 write units = 5KB/second
  - How much data can be read in eventually consistent manner?
    - 4 KB * 2 eventually consistent reads * 5 read units = 40 KB/second
  - How much data can be read in strongly consistent manner?
    - 4 KB * 1 strongly consistent read * 5 read units = 20 KB/second

Example

- How many read units are required if we want to sustain the throughput capacity for eventually consistent reads of 120KB/second?
  - 4 KB Chunks to read = 120 / 4 = 30
  - 1 read unit allows 2 eventually consistent reads of 4 KB chunks each
  - Number of read units for 30 chunks = 30 / 2 = 15

Best Practice for Tables

- How to choose Partition keys?
  - Goal: Spread the access uniformly across Primary key partition space
  - DynamoDB divides a table's items into multiple partitions, and distributes the data primarily based upon the partition key value.

- Partition key value – Uniformity
  - User ID, where the application has many users – Good
  - Status code, where there are only a few possible status codes. – Bad
  - Item creation date, rounded to the nearest time period (e.g. day, hour, minute) – Bad
  - Device ID, where each device accesses data at relatively similar intervals – Good
  - Device ID, where even if there are a lot of devices being tracked, one is by far more popular than all the others. – Bad

Best Practice for Items

- Use One-to-Many Tables instead of Large Set Attributes
  - Forum → Thread → Reply
  - A Forum has multiple threads
  - A Thread has multiple replies
- Design 1
  - Single ThreadForum table with 'Set' attribute for Replies for each thread
  - Problems:
    - All Replies for a thread will get shipped around always (as part of Thread get/put). This will cost throughput capacity units
    - Difficult to query for only certain replies (within some date range)
    - The size of an attribute cannot exceed certain limit (400 KB)
- Design 2: Alternate (and better) design
  - Separate Thread and Reply tables
  - Each Item (row) in the Reply table needs to maintain a 'pointer' to the Primary key Id of the Item in the Thread table to which that Reply belongs
    - This needs to be done at the application level

Best Practice for Items

- Compress Large Attribute Values
  - This saves throughput capacity units
- Store Large Attribute Values in S3
  - E.g.: Binary objects such as images
- Break up Large Attributes Across Multiple Items
  - E.g.: Break contents of a Reply into multiple chunks

DynamoDB Pricing

- Free Tier
  - 25 write capacity units
  - 25 read capacity units
- Possible number of requests in a month
  - (25 writes + 50 eventually consistent reads)/second * 60 seconds * 60 minutes * 24 hours * 31 days
  - 75 * 60 * 60 * 24 * 31
  - 200880000
  - 200 Million requests / month

Secondary Indexes

- A Secondary Index is like a “copy” of table with only selected attributes
  - Default: The secondary index attributes and the primary key attributes of the table
- A secondary index allows querying the table using an alternate key in addition to using the primary key
- Global secondary index
  - An index with a partition key and sort key that can be different from those on the table
  - Up to 5 per table
- Local secondary index
  - An index that has the same partition key as the table, but a different sort key
  - Up to 5 per table

Secondary Indexes

{
“**PersonID**”: 101,
“LastName”: “Smith”,
“FirstName”: “Fred”,
“Phone”: “555-4321”
}

{
“**PersonID**”: 102,
“LastName”: “Jones”,
“FirstName”: “Mary”,
“Phone”: “555-4321”
}

Person Table

{
“PersonID”: 101,
“**LastName**”: “Smith”,
“**FirstName**”: “Fred”,
}

{
“PersonID”: 102,
“**LastName**”: “Jones”,
“**FirstName**”: “Mary”,
}

Secondary Index

DynamoDB Streams

- Capture Data Modification events in DynamoDB tables
- Need to be enabled on a table
- Each event is represented by a stream record
- If a new item is added to the table, the stream captures the new item (all attributes)
- If an item is updated, the stream captures “before” and “after” of any attributes that were modified
- If an item is deleted, the stream captures entire item before it was deleted

Streams API

- ListStreams
- DescribeStream
- GetShardIterator
  - Data structure that your application uses to retrieve records from the stream
- GetRecords
  - Retrieves one or more stream records using a given shard iterator

Authentication and Access Control

- Resources
  - Table
    - Index
    - Stream
- Table
  - arn:aws:dynamodb:region:account-id:table/table-name
- Index
  - arn:aws:dynamodb:region:account-id:table/table-name/index/index-name
- Stream
  - arn:aws:dynamodb:region:account-id:table/table-name/stream/stream-label

- Index and Stream are defined as sub-resources of the Table resource

Policies

- DynamoDB only supports IAM Policies

- DynamoDB *does not* support resource-based policies

Resource Ownership

- If you use the root account credentials of your AWS account to create a table, your AWS account is the owner of the resource (in DynamoDB, the resource is a table).
- If you create an IAM user in your AWS account and grant permissions to create a table to that user, the user can create a table. However, your AWS account, to which the user belongs, owns the table resource.
- If you create an IAM role in your AWS account with permissions to create a table, anyone who can assume the role can create a table. Your AWS account, to which the user belongs, owns the table resource.

Access Control

- How to define policies for Tables/Indexes?

- How to define fine-grained policies?
  - Per Primary Key
  - Per Attribute

IAM Policy: Example 1

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllAPIActionsOnBooks",
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Allow a User to Perform *Any* DynamoDB Actions on a Table

Example 2

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAPIActionsOnBooks",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Allow *Read-only Access* on Items in a Table

Example 3

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutUpdateDeleteOnBooks",
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Allow Put, Update, and Delete Operations on a Specific Table

Example 4

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListTables",
            "Effect": "Allow",
            "Action": [
                "dynamodb:ListTables"
            ],
            "Resource": "*"
        }
    ]
}
```

The Wildcard (*) in the Resource value means that you can use this action to obtain the names of all the tables owned by the AWS account in the current AWS region

Example 5

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessAllIndexesOnBooks",
            "Effect": "Allow",
            "Action": [
                "dynamodb:*"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books",
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books/index/*"
            ]
        }
    ]
}
```

Allow Access to a Specific Table and All its Indexes

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllAPIActionsOnUserSpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:*"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/$
{aws:username}_ProductCatalog"
        },
        {
            "Sid": "AdditionalPrivileges",
            "Effect": "Allow",
            "Action": [
                "dynamodb:ListTables",
                "dynamodb:DescribeTable",
                "cloudwatch:*",
                "sns:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy variable used so that this policy can be attached to different IAM users

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReservedCapacityDescriptions",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeReservedCapacity",
                "dynamodb:DescribeReservedCapacityOfferings"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:*"
        },
        {
            "Sid": "DenyReservedCapacityPurchases",
            "Effect": "Deny",
            "Action": "dynamodb:PurchaseReservedCapacityOfferings",
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:*"
        }
    ]
}
```

Prevent a User from Purchasing Reserved Capacity Offerings

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessGameScoresStreamOnly",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeStream",
                "dynamodb:GetRecords",
                "dynamodb:GetShardIterator",
                "dynamodb:ListStreams"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/stream/*"
        }
    ]
}
```

Allow Read Access for a DynamoDB Stream Only

Fine-grained Access Control Policies

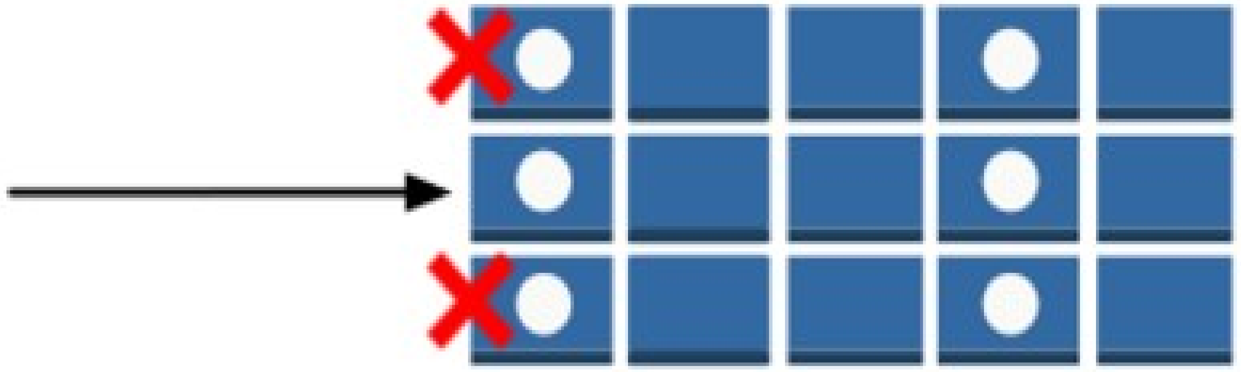
- Using Condition definition in IAM Policy
  - Allow users read-only access to certain items and attributes in a table/secondary index
  - Allow users write-only access to certain items and attributes based upon the identity of the user

- https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/specifying-conditions.html

Grant access to specific 'row'

Grant permissions on a table, but restrict access to specific items in that table based on certain primary key values.

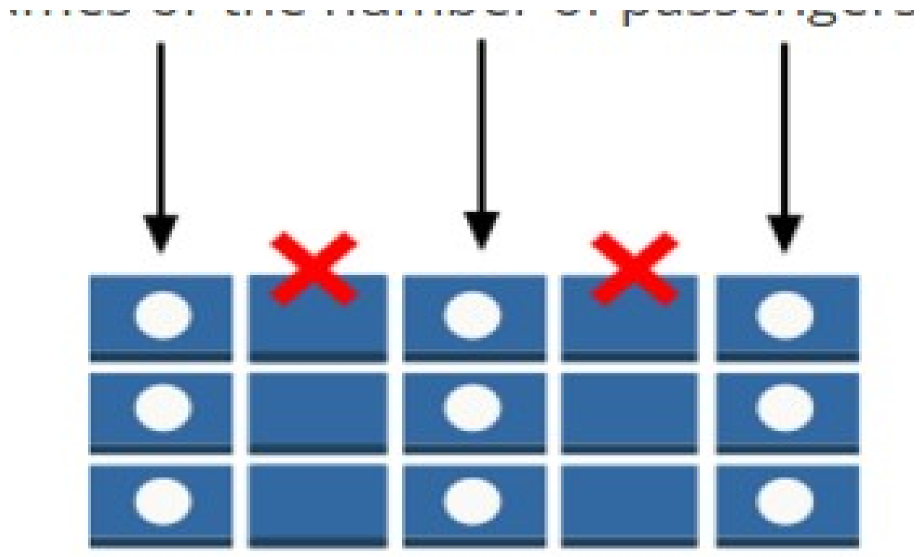
Example - Social networking app for games, where all users' saved game data is stored in a single table, but users should not access data items that they do not own



Grant access to specific 'columns'

Hide information so that only a subset of attributes are visible to the user.

Example - An app that displays flight data for nearby airports based on the user's location. Airline names, arrival and departure times, and flight numbers are all displayed. However, attributes such as pilot names or the number of passengers are hidden.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToUserItems",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": [
                        "${www.amazon.com:user_id}"
                    ]
                }
            }
        }
    ]
}
```

Grant read/write access to a user for only his/her record in the GameScores table

LeadingKeys → Primary Key (Hash key)

ForAllValues

ForAllValues – The condition returns true if there's a match between *every one* of the specified key values in the request and at least one value in the policy.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "GetItem",
    "Resource": "arn:aws:dynamodb:REGION:ACCOUNT-ID-WITHOUT-HYPHENS:table/Thread",
    "Condition": {"ForAllValues:StringLike": {"dynamodb:requestedAttributes": [
      "PostDateTime",
      "Message",
      "Tags"
    ]}}
  }]
}
```

The GetItem action on Thread table will be allowed if the request is for one of PostDateTime, Message, Tags attributes.

If the request included ID attribute as well, it will be denied because ID is not specified as one of the requestAttributes, and ForAllValues requires every one of the request attribute to match at least one of the attribute names in the policy

ForAnyValues

ForAnyValue – The condition returns true if any one of the key values in the request matches any one of the condition values in the policy.

```
{
   "Version": "2012-10-17",
   "Statement": [{
            "Effect": "Deny",
            "Action": "PutItem",
            "Resource": "arn:aws:dynamodb:REGION:ACCOUNT-ID-WITHOUT-HYPHENS:table/Thread",
           "Condition": {"ForAnyValue:StringLike": {"dynamodb:requestedAttributes": [
                   "ID",
                   "PostDateTime"
           ]}}
    }]
}
```

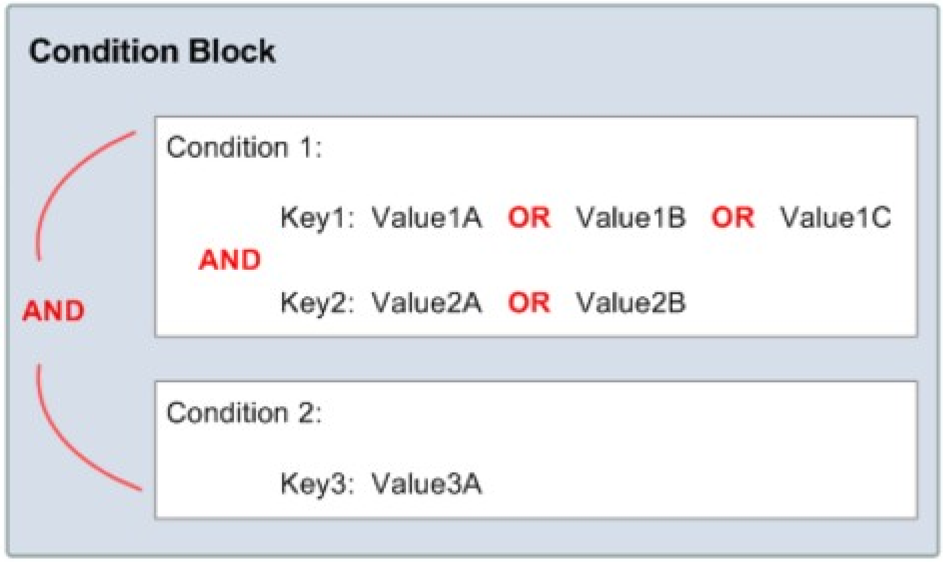
Deny PutItem action to Thread table if the action contains updates to either ID attribute or PostDateTime attribute

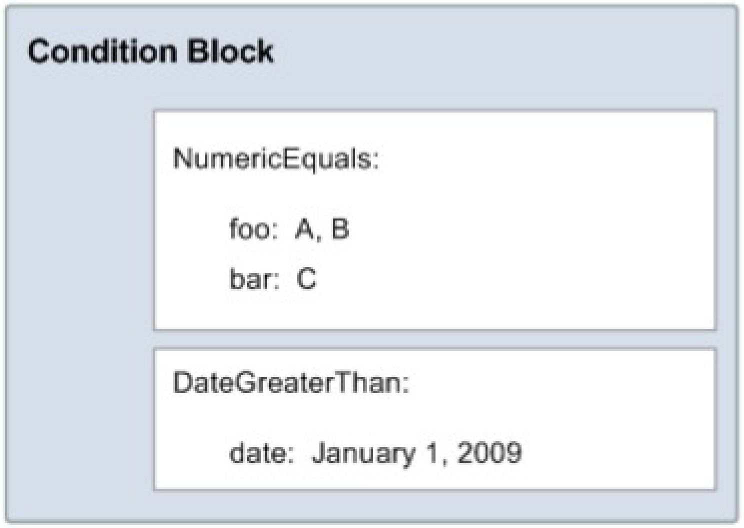
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_multi-value-conditions.html

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "LimitAccessToSpecificAttributes",
         "Effect": "Allow",
         "Action": [
            "dynamodb:UpdateItem",
            "dynamodb:GetItem",
            "dynamodb:Query",
            "dynamodb:BatchGetItem",
            "dynamodb:Scan"
         ],
         "Resource": [
            "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores"
         ],
         "Condition": {
            "ForAllValues:StringEquals": {
               "dynamodb:Attributes": [
                  "UserId",
                  "TopScore"
               ]
            },
            "StringEqualsIfExists": {
               "dynamodb:Select": "SPECIFIC_ATTRIBUTES",
               "dynamodb:ReturnValues": [
                  "NONE",
                  "UPDATED_OLD",
                  "UPDATED_NEW"
               ]
            }
         }
      }
   ]
}
```

Grant read/write access to only specific attributes in the GameScores table if
- The requested attribute is either UserId OR TopScore AND
- (The action is select with specific_attributes as the return Clause AND
- ReturnValues for UpdateItem is set to either NONE, UPDATED_OLD, UPDATED_NEW)

IAM Condition Block





https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition.html

IfExists Condition Operators

- What does this mean?
  - If the key is present in the context of the request, process the key as specified in the policy.
  - If the key is not present, evaluate the condition element as true.

- Example:
  - Launching of EC2 instance
  - https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html#Conditions_IfExists

IfExists Condition Operators

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "THISPOLICYDOESNOTWORK",
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {"StringLike": {"ec2:InstanceType": [
      "t1.*",
      "t2.*",
      "m3.*"
    ]}}
  }
}
```

This policy will cause failure when launching EC2 instances. This is because when launching a EC2 instance several other resources are required such as Images, Security groups etc. The Condition will be evaluated against all these resources. And they won't have the ec2:InstanceType attribute defined so the policy will evaluate to false.

IfExists Condition Operator

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {"StringLikeIfExists": {"ec2:InstanceType": [
      "t1.*",
      "t2.*",
      "m3.*"
    ]}}
  }
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QueryOnlyProjectedIndexAttributes",
            "Effect": "Allow",
            "Action": [
                "dynamodb:Query"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-
2:123456789012:table/GameScores/index/TopScoreDateTimeIndex"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:Attributes": [
                        "TopScoreDateTime",
                        "GameTitle",
                        "Wins",
                        "Losses",
                        "Attempts"
                    ]
                },
                "StringEquals": {
                    "dynamodb:Select": "SPECIFIC_ATTRIBUTES"
                }
            }
        }
    ]
}
```

Query specific attributes from the TopScoreDateTimeIndex

DynamoDB API Permissions

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/api-permissions-reference.html

IAM Policies for DynamoDB

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/using-identity-based-policies.html

Python Programming

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.01.html

IaaS and IaC

Devdatta Kulkarni

Infrastructure-as-a-Service (IaaS)

- What is Infrastructure?

- What is 'as-a-service' model?

What is Infrastructure?

- Compute
  - Virtual Machines (EC2)
  - Containers (Docker)
- Storage
  - Object Storage (S3)
  - Non-relational Stores (DynamoDB)
  - Relational Stores (Amazon RDS, Google Cloud SQL)
- Networking
  - Virtual Private Cloud (VPC)
  - Load Balancers (AWS ELB)

“as-a-service” Model Characteristics

- A 'web service' running on public / private data centers and accessible over HTTP
- Typically exposing APIs (REST, SOAP)
  - Client-side tools include CLIs, SDKs, UI
- Multi-tenant
  - Same service is able to support multiple users (“clients/customers”)
- Pay-as-you use charging model

Infrastructure-as-a-Service (IaaS)

- What is Infrastructure-as-a-Service (IaaS)?
  - A web service that supports provisioning and management of infrastructure resources (compute, storage, networking, etc.)
    - EC2 is IaaS for VMs
    - S3 is IaaS for Object Storage
    - DynamoDB is IaaS for Non-relational Datastores
    - RDS is IaaS for Relational Databases

Infrastructure Provisioning & Management

- Provisioning & Management
  - Create, Update, Manage (Taking backup, Restoring, Changing security policies, etc.)
- Infrastructure provisioning in real world
  - Does not happen through direct calls to IaaS systems as we have done so far!!
  - Typically done through tools that support a 'declarative' model for provisioning and management
    - AWS CloudFormation, Hashicorp Terraform

AWS CloudFormation

- Create a Template describing the Infrastructure resources that you want
  - YAML, JSON format
    - This is the 'declarative definition'
- Give the template to CloudFormation web service
  - It takes care of provisioning all the infrastructure resources defined in the template
- You don't have to write code for provisioning
  - Like you did for assignment1 for provisioning EC2 instance
  - Or assignment2 for creating S3 Bucket

Example: CF Template to Provision a EC2 instance (“stack”)

```
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
Resources:
  MyEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-2f726546"
      InstanceType: t1.micro
      KeyName: testkey
      BlockDeviceMappings:
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

Provision a EC2 Instance of the given InstanceType with specified AMI and an EBS Block Device Mapping

Is it a Service? Is it Something else?

- Is AWS CloudFormation a IaaS?

- New term to describe systems like CloudFormation:
  - Infrastructure-as-Code

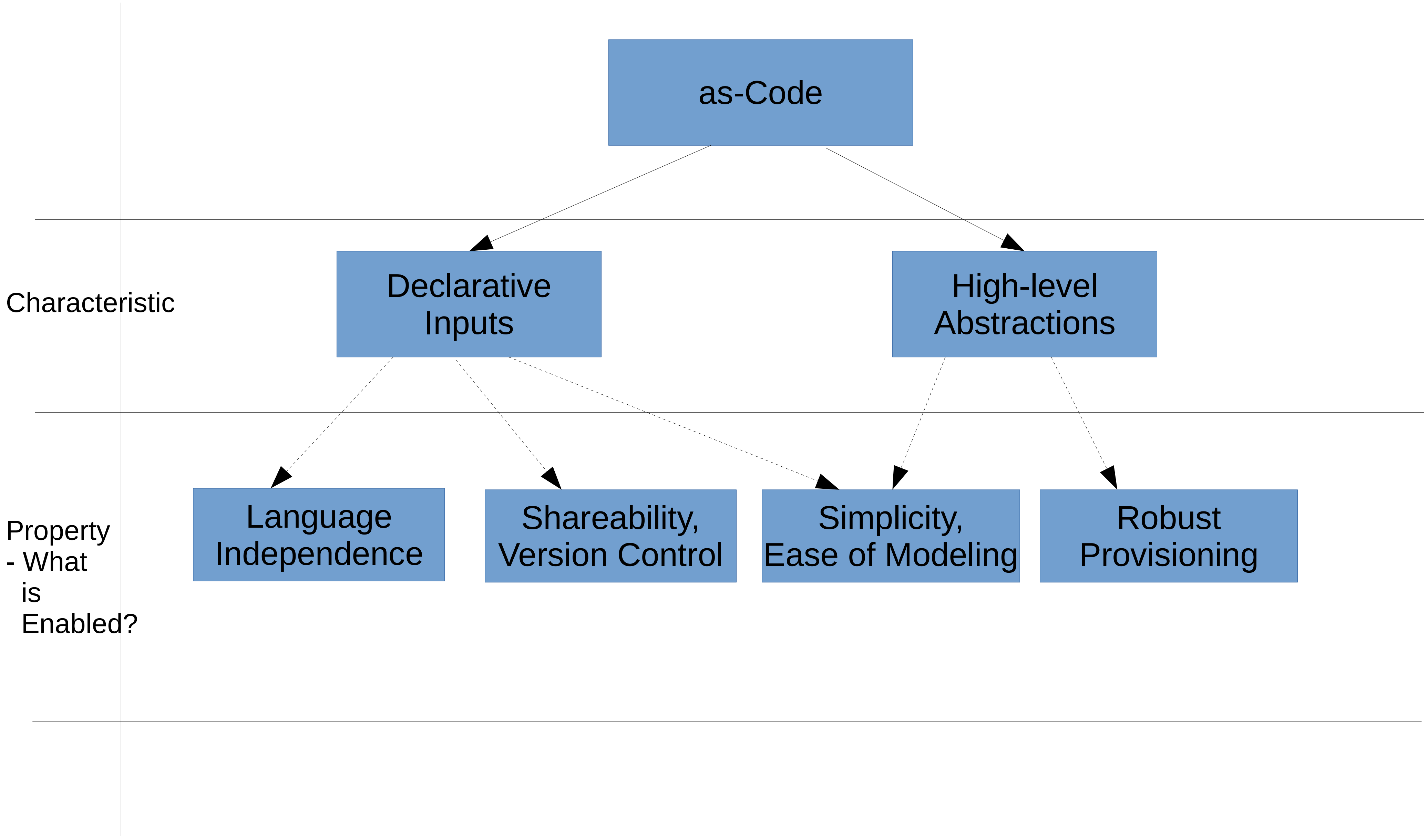
- Technically IaC system is:
  - A web service that works with infrastructure defined in declarative format
    - This declarative definition is called as the infrastructure's “code”
      - The declarative definition “codifies” infrastructure

Characteristics of “as-Code” systems

- Declarative Inputs
  - Express what is required, not how to do it
    - Example of Declarative input: SQL
      - select * from Movies where year = '2010';

- High-level Abstractions
  - Resource, ResourceGroups, etc.

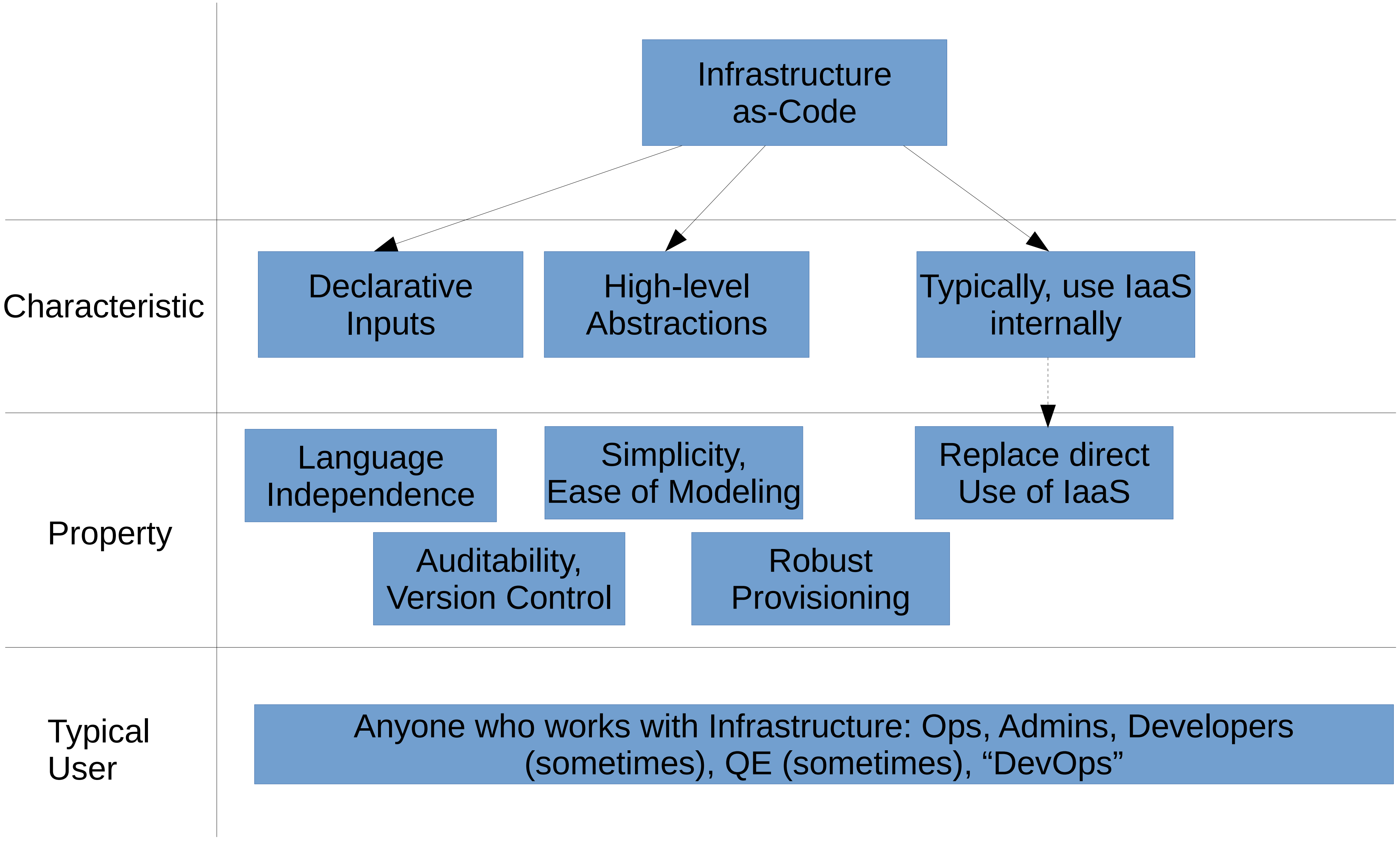
“as-Code” characteristics



Advantages of “as-Code” Model

- Easier to understand what is being provisioned
  - With procedural model (like our programs) this is difficult as 'what' is being provisioned is embedded in the program code
- Robust provisioning of Infrastructure
  - Single resource provisioning
    - Service like CloudFormation will typically include robust retry logic to handle single resource provisioning failures
  - Multi-resource provisioning
    - Multiple resources will be treated as a single unit from provisioning point-of-view
- Shareable Infrastructure definition
  - Infrastructure definition templates can be shared easily with others
- Repeatable creation of Infrastructure
  - Easy to re-create resource stacks corresponding to the defined infrastructure resource template
  - Easy to rollback to previous version of a resource
- Auditability of the Infrastructure
  - Easy to track Infrastructure resource templates in a version control system (github, bitbucket, etc.)

Infrastructure-as-Code (IaC) Characteristics

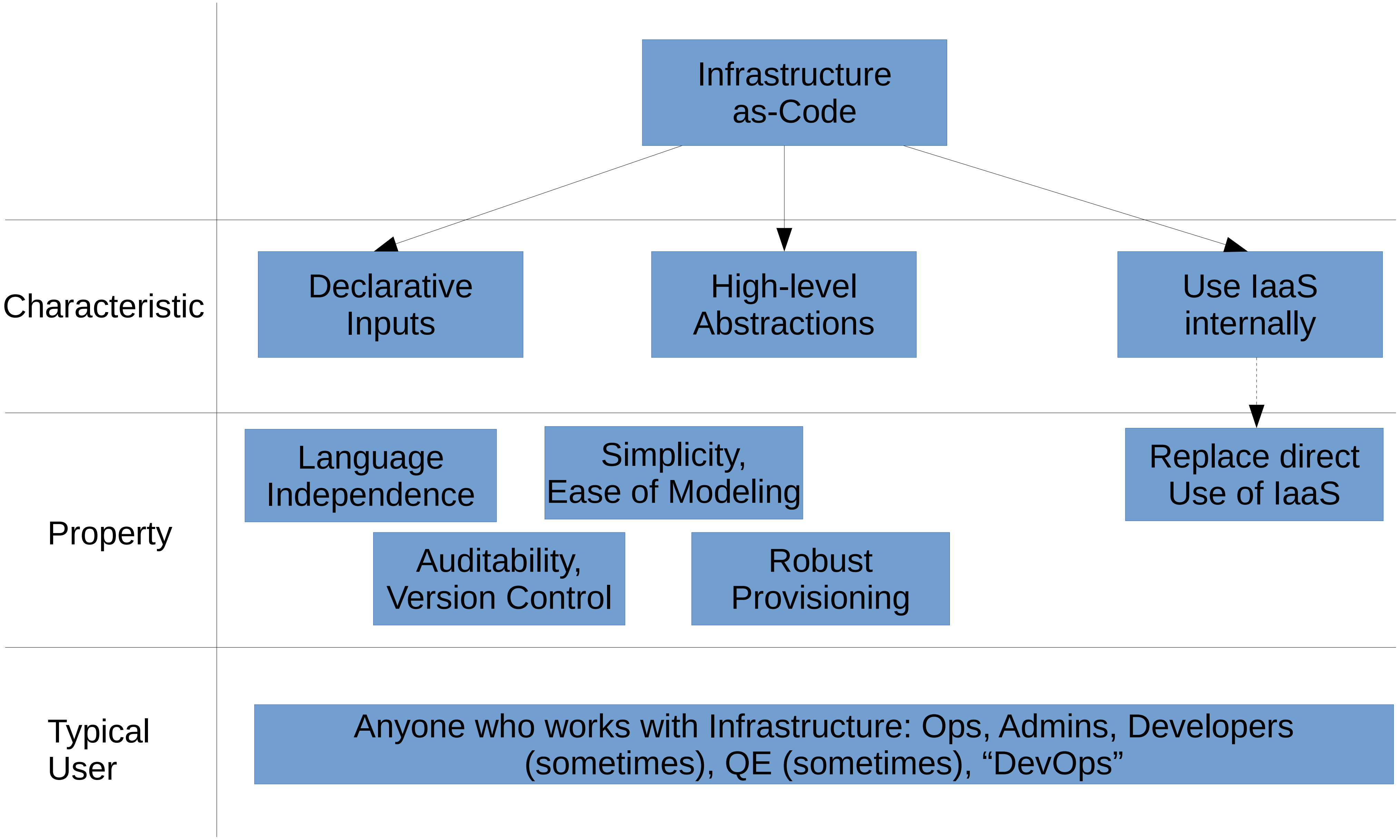

Infrastructure as-Code
Characteristic
Declarative Inputs
High-level Abstractions
Typically, use IaaS internally
Property
Language Independence
Simplicity, Ease of Modeling
Replace direct Use of IaaS
Auditability, Version Control
Robust Provisioning
Typical User
Anyone who works with Infrastructure: Ops, Admins, Developers (sometimes), QE (sometimes), “DevOps”

Infrastructure-as-Code

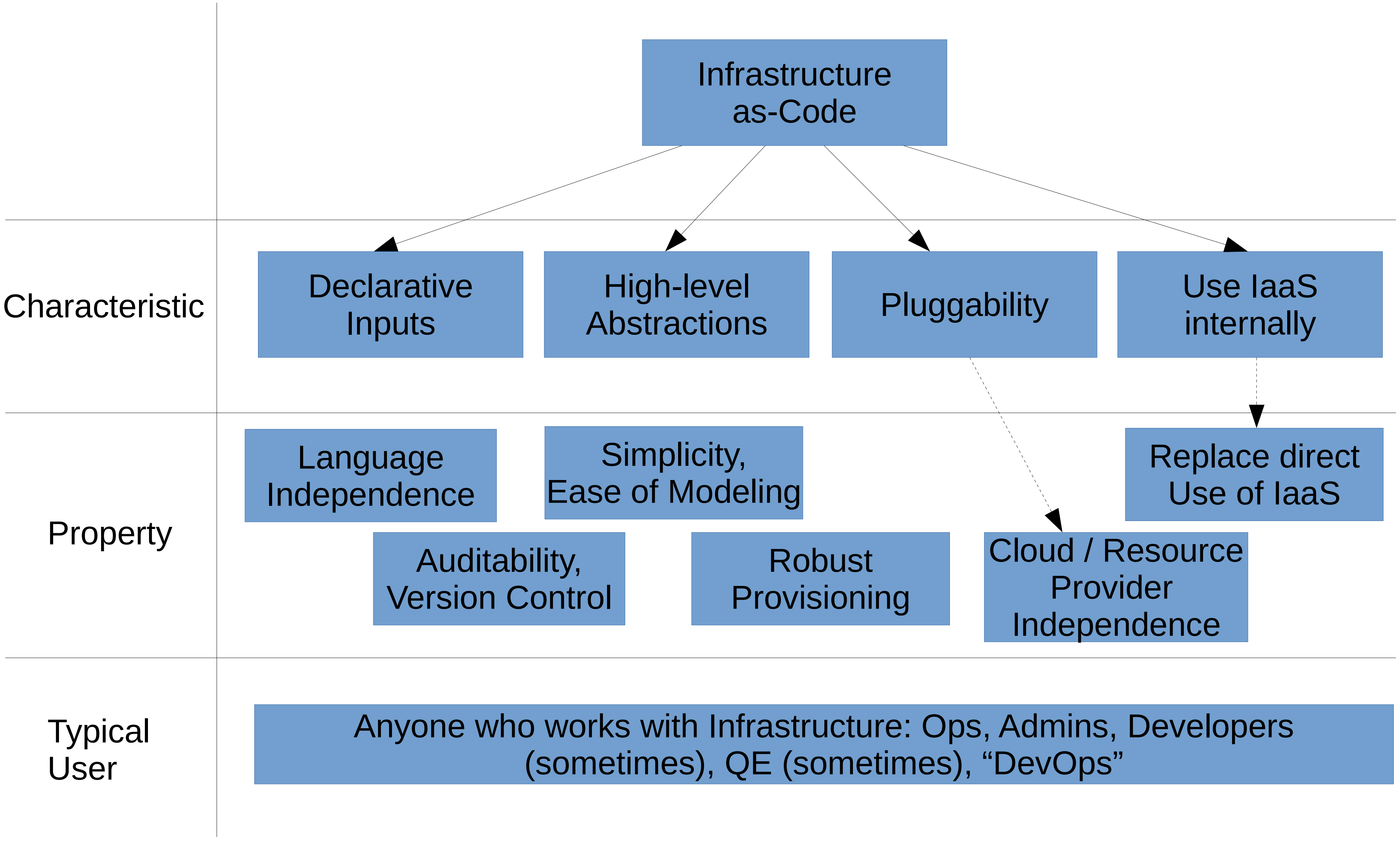
- High-level abstractions model 'Infrastructure resource'
  - VM, Database, Bucket, LoadBalancer, etc.

- Who is the target user?
  - Person(s) who has responsibility of managing the infrastructure
    - Ops, DevOps, SysAdmin, Site Reliability Engineer (SRE)
  - Typical skills needed
    - Deep understanding of various resources, their properties, their life-cycle actions
    - Scripting – Bash, Python
      - Create, manipulate infrastructure template files
      - Make calls to CloudFormation/Terraform through their CLI

AWS CloudFormation Characteristics

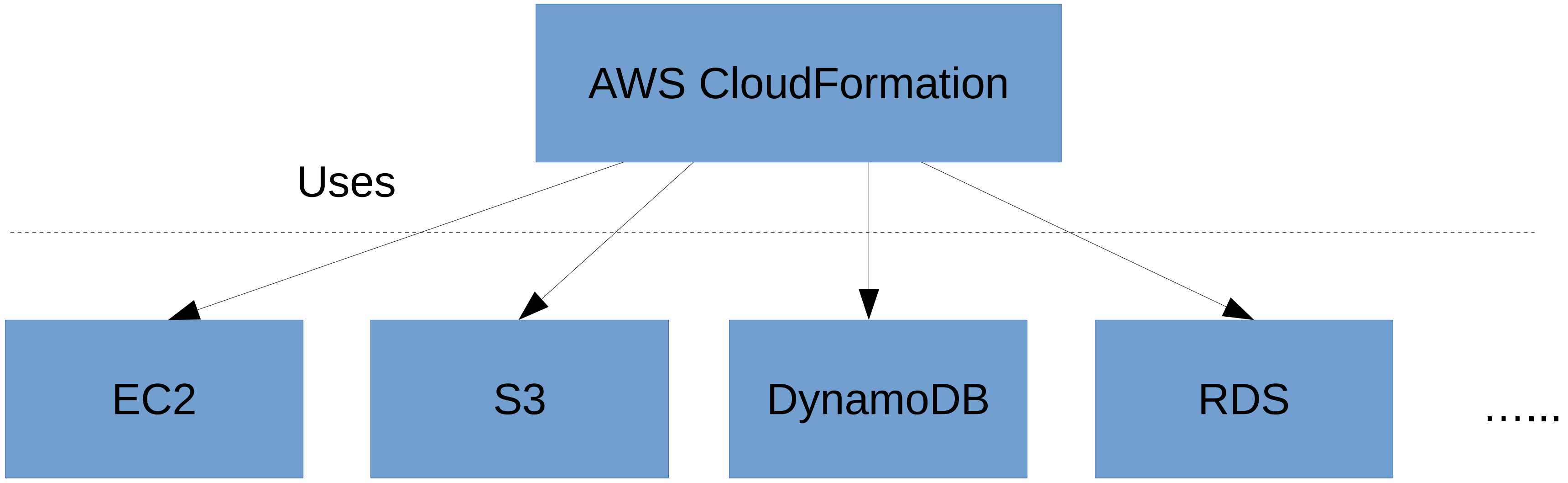

Infrastructure as-Code
Characteristic
Declarative Inputs
High-level Abstractions
Use IaaS internally
Property
Language Independence
Simplicity, Ease of Modeling
Replace direct Use of IaaS
Auditability, Version Control
Robust Provisioning
Typical User
Anyone who works with Infrastructure: Ops, Admins, Developers (sometimes), QE (sometimes), “DevOps”

Terraform IaC Characteristics


Infrastructure as-Code
Characteristic
Declarative Inputs
High-level Abstractions
Pluggability
Use IaaS internally
Property
Language Independence
Simplicity, Ease of Modeling
Replace direct Use of IaaS
Auditability, Version Control
Robust Provisioning
Cloud / Resource Provider Independence
Typical User
Anyone who works with Infrastructure: Ops, Admins, Developers (sometimes), QE (sometimes), “DevOps”

AWS CloudFormation

- https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html



Concepts

- Abstraction
  - Resources
- Templates
  - JSON or YAML formatted text file that declares/defines one or more resources
- Stack
  - The set of resources that are provisioned

What is Declared? What is not?

- Declared
  - Resources with their different Properties
  - Input Parameters
  - Output Parameters
- Not declared
  - AWS IAM Permissions
    - Stack creation will fail if you don't have appropriate permissions for all the resources that are defined in the template

Declarative System Challenges

- How to make a template reusable?
  - In CF, achieved through Input parameters, mappings
    - In AWS IAM there are Policy variables for this purpose
  - Include one template in another template?
- How to define “ordering” between different resource's provisioning?
  - Explicit specification
  - Implicitly defined
    - In CF, implicitly defined through the 'DependsOn' attribute
- Are all life-cycle actions of a resource supported?
  - Create, Read, Update, Delete
  - Setup/Configure, Take backup, etc.
- How to define resource delete action?
  - In OpenStack Heat, delete is achieved through removing a resource from a template and doing stack update on an existing stack

Declarative System Challenges

- Semantics
  - All-or-Nothing
    - Either all resources are created successfully or None are created
    - What happens if provisioning action fails after creating some resources in the template?
      - All-or-Nothing semantics require you to delete the successfully created resources!!
  - Partially created
    - Is this useful in any situation?

CFN Template

```
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

$\xrightarrow{}$ Required; rest are optional

Resources

- Top-level Object in a CF Template

```
Resources:
  HelloBucket:            ──────────► Logical name
    Type: AWS::S3::Bucket
```

A S3 Bucket resource with the name HelloBucket

Logical name & Physical name

- Logical name
  - Name that is used in the template
- Physical name
  - Combination of Logical Name, Stack Name, Unique ID
  - AWS generates the physical name

Resource Properties

- Specified with the resource

```
Resources:
  HelloBucket:
    Type: AWS::S3::Bucket
    Properties:
      AccessControl: PublicRead
```

HelloBucket S3 bucket with canned ACL (PublicRead)

Resource Properties

- There can be multiple properties
- A Property can have multiple values

```
Resources:
  HelloBucket:
    Type: AWS::S3::Bucket
    Properties:
      AccessControl: PublicRead
      WebsiteConfiguration:
        IndexDocument: index.html
        ErrorDocument: error.html
```

Resource Types

- Almost all AWS resources
- https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html
- https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-product-property-reference.html

Resource Attributes

- CreationPolicy
- DeletionPolicy
- DependsOn
- Metadata
- UpdatePolicy
  - Only applies to AWS::AutoScaling::AutoScalingGroup and AWS::Lambda::Alias

CreationPolicy

This attribute when associated with a resource prevents its status from reaching create complete until CloudFormation receives specified number of success signals or the timeout period is exceeded

```
CreationPolicy:
  AutoScalingCreationPolicy:
    MinSuccessfulInstancesPercent: Integer
  ResourceSignal:
    Count: Integer
    Timeout: String
```

DeletionPolicy

- This attribute allows preserving the state of a resource before it is deleted
  - E.g.: Taking backup
- You can also specify whether the resource should be kept around even if the stack is deleted
  - This is done through a 'Retain' policy
  - AWSTemplateFormatVersion: '2010-09-09'

    ```
    Resources:
      myS3Bucket:
        Type: AWS::S3::Bucket
        DeletionPolicy: Retain
    ```
  - DeletePolicy Options
    - Delete (This is the default if no DeletePolicy option is specified for a resource)
    - Retain
    - Snapshot

DependsOn

- With the DependsOn attribute you can specify that the creation of a specific resource follows another.
- When you add a DependsOn attribute to a resource, that resource is created only after the creation of the resource specified in the DependsOn attribute.

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    DependsOn: myDB
  myDB:
    Type: AWS::RDS::DBInstance
    Properties:
      AllocatedStorage: '5'
      DBInstanceClass: db.m1.small
      Engine: MySQL
      EngineVersion: '5.5'
      MasterUsername: MyName
      MasterUserPassword: MyPassword
```

Metadata

The Metadata attribute enables you to associate structured data with a resource.

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Metadata:
      Object1: Location1
      Object2: Location2
```

Intrinsic Functions

- Provided by AWS
- Use these in your templates to assign values to properties that are not available until runtime
- Can be used in following locations in the template
  - Resource Properties
  - Outputs
  - Metadata attributes
  - Update Policy attributes

Ref

- In-built function available in AWS
- Input
  - Logical name of resource/parameter
- Output
  - Physical name of resource/parameter

Ref

```
Resources:
  Ec2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      SecurityGroups:
        - !Ref InstanceSecurityGroup
        - MyExistingSecurityGroup
      KeyName: mykey
      ImageId: ami-7a11e213
  InstanceSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIp: 0.0.0.0/0
```

- Interlinking of resources
- Literal names can be used as Values
- Key with name mykey needs to exist in your account
- MyExistingSecurityGroup needs to exist

Fn:GetAtt

- In-built function to get attributes of a resource
- Input
  - Logical name of the resource
  - Name of the attribute to be retrieved
- Output
  - Attribute value

Fn:GetAtt

```
Resources:
  myBucket:
    Type: 'AWS::S3::Bucket'
  myDistribution:
    Type: 'AWS::CloudFront::Distribution'
    Properties:
      DistributionConfig:
        Origins:
          - DomainName: !GetAtt
              - myBucket
              - DomainName
            Id: myS3Origin
            S3OriginConfig: {}
        Enabled: 'true'
        DefaultCacheBehavior:
          TargetOriginId: myS3Origin
          ForwardedValues:
            QueryString: 'false'
          ViewerProtocolPolicy: allow-all
```

Get the DomainName attribute
of the Bucket 'myBucket'

Configure the CloudFront Distribution
to use this Bucket

Input Parameters

- A parameter definition consists of
  - Type of the parameter
    - String, Number, or an AWS-specific type
  - Constraints on the parameter value
  - Any Default value

Parameters

```
Parameters:
  KeyName:
    Description: The EC2 Key Pair to allow SSH access to the instance
    Type: 'AWS::EC2::KeyPair::KeyName'
Resources:
  Ec2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      SecurityGroups:
        - !Ref InstanceSecurityGroup
        - MyExistingSecurityGroup
      KeyName: !Ref KeyName
      ImageId: ami-7a11e213
  InstanceSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIp: 0.0.0.0/0
```

Platforms and PaaS

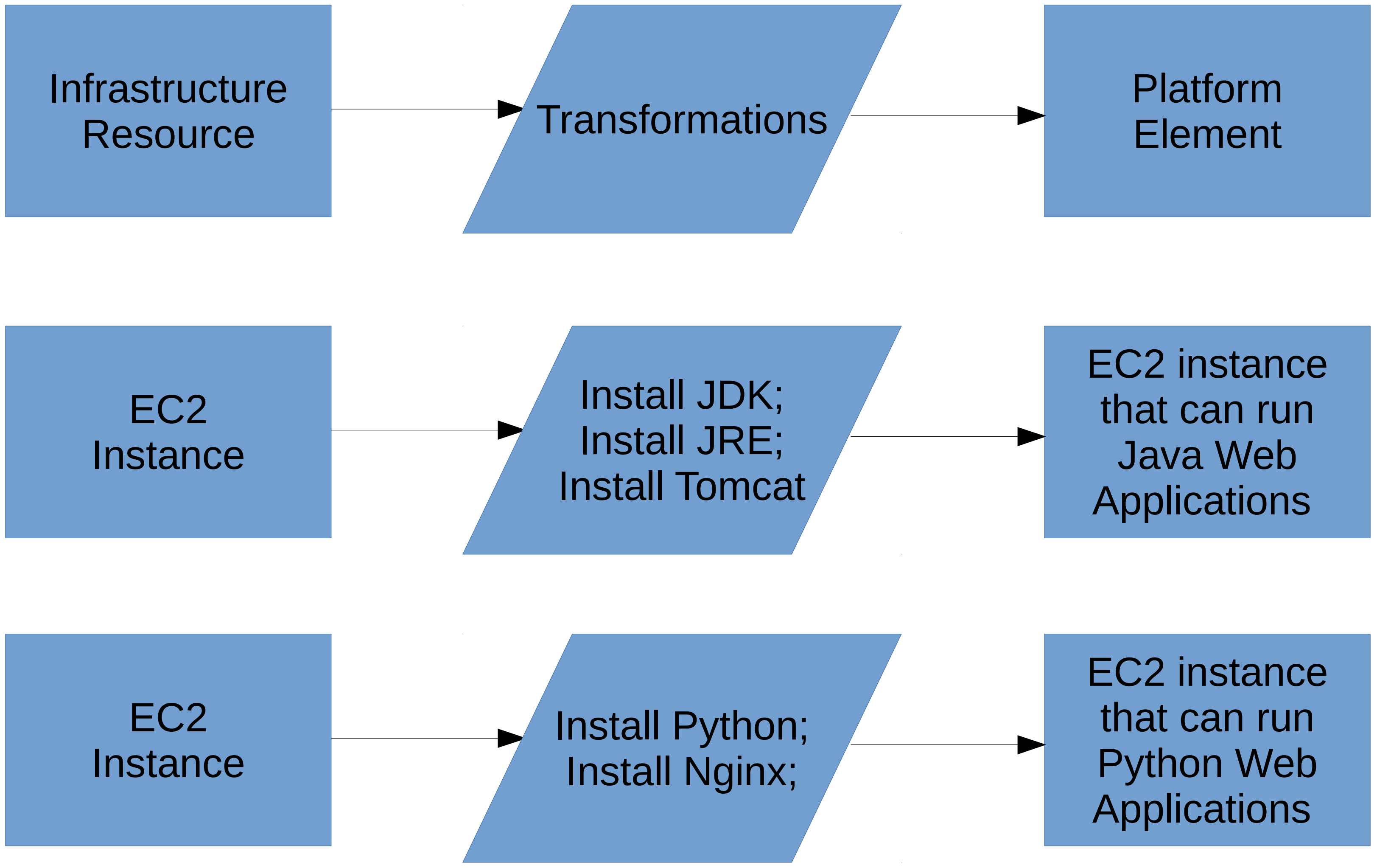
Devdatta Kulkarni

What is Platform?

- Platform is a set of one or more Infrastructure resources that are configured to run specific kind of applications

- Java Web Application Platform
  - A EC2 instance that is configured with JDK, JRE, Apache Tomcat to run Java Web Applications
  - A EC2 instance like above + DynamoDB table appropriately configured
  - 2 EC2 instances like above + S3 bucket + DynamoDB table + RDS database instance
  - Etc.
- Python Web Application Platform
  - A EC2 instance that is configured with Python2.7 and a web server like Nginx to run Python Applications

Infrastructure -> Platform

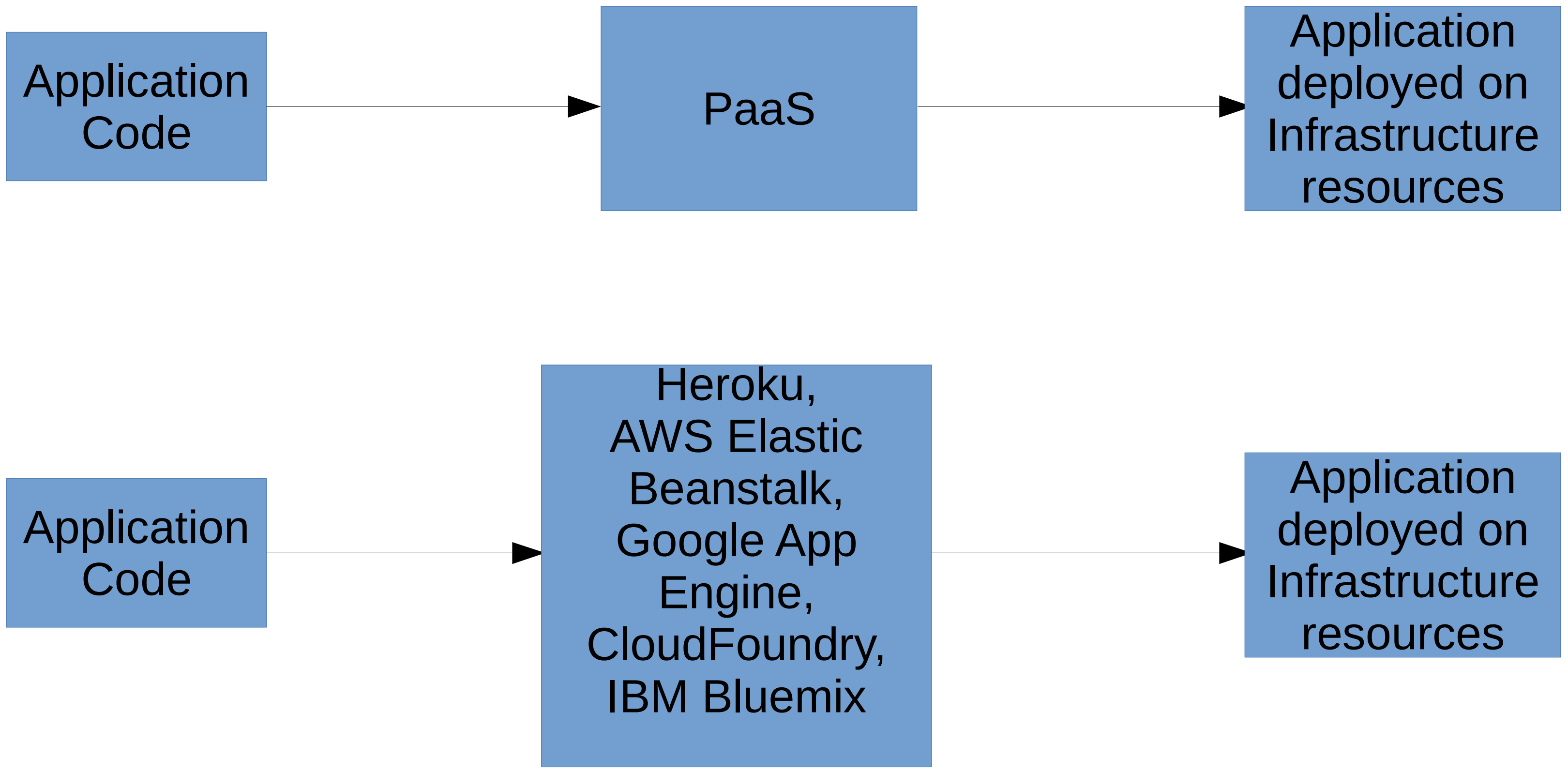

Infrastructure Resource
Transformations
Platform Element
EC2 Instance
Install JDK; Install JRE; Install Tomcat
EC2 instance that can run Java Web Applications
EC2 Instance
Install Python; Install Nginx;
EC2 instance that can run Python Web Applications

Platform-as-a-Service

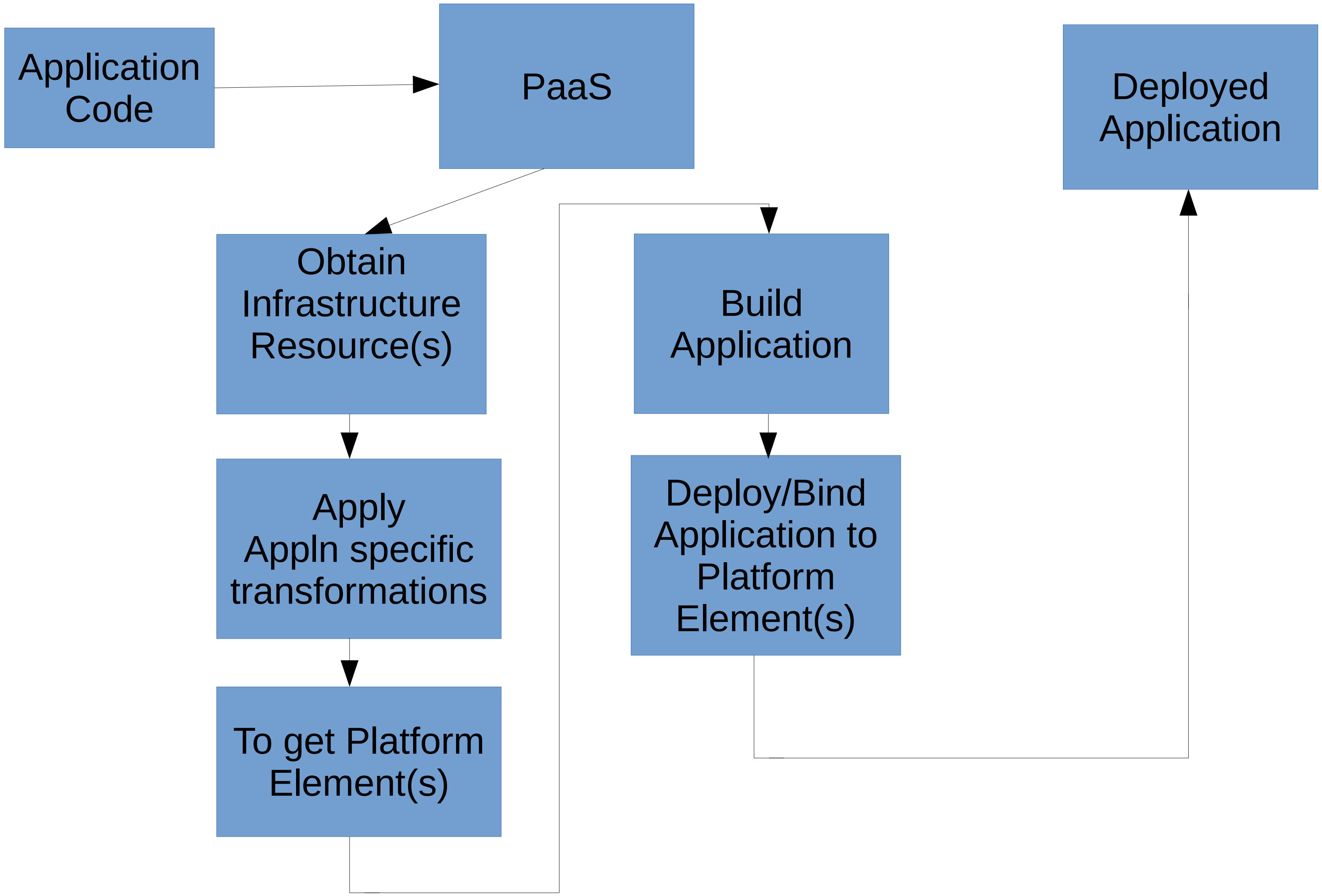
- A web service that applies transformations on infrastructure resources to generate the required Platform

- Current popular PaaSes don't stop there though
- They also deploy application on the generated Platform
  - Is this intrinsic to defining Platform-as-a-Service?

Current PaaS Model


Application Code
PaaS
Application deployed on Infrastructure resources
Application Code
Heroku, AWS Elastic Beanstalk, Google App Engine, CloudFoundry, IBM Bluemix
Application deployed on Infrastructure resources

Current Popular PaaS Model


Application Code
PaaS
Deployed Application
Obtain Infrastructure Resource(s)
Build Application
Apply Appln specific transformations
Deploy/Bind Application to Platform Element(s)
To get Platform Element(s)

IaaS vs. PaaS

- IaaSes deal with infrastructure resources
- IaaS systems create infrastructure resources
- Typically focus on *create/delete operations* of infrastructure resource
- User needs to do resource *configuration* separately
  - Using configuration management tools like Chef, Puppet
- Relatively easy to understand if resource creation fails

- PaaSes deal with applications
- PaaS systems create running application
- Performs several steps
  - Creating required infrastructure resources
  - Transforming/
    Configuring the resources
  - Building the application code
  - Deploying it on configured infrastructure resources
- Typically complex to decipher encountered errors/issues
- PaaS systems typically use IaaS/IaC systems internally

What about IaC?

- How does Infrastructure-as-Code fit into the equation?
  - Some of the PaaS actions can be done using IaC features

Overlapping capabilities

- IaC systems can support some of the steps done by PaaS
  - E.g.: We can define a CloudFormation template that pulls application code from github, builds it, and deploys it
  - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/deploying.applications.html
- Should we use IaC for deploying web applications then?
  - May be for pre-built/static applications
  - In general case no. Why?
    - IaC systems are optimized for working with *infrastructure*
    - The target user persona for IaC systems is a Operations Person (Ops Engineer, DevOps Engineer) and not Application developer

Use IaC to deploy web applications?

- May be Ok for 'pre-built' applications
  - Wordpress blog, Moodle e-learning web application, etc.
  - Why?
    - Application code is 'static' – it is not going to change ever (e.g.: Wordpress code)
      - So Operators can define a template for application deployment once and then use it often
- For custom applications
  - Application code is under constant change - application developers are actively developing their code every day
  - IaC systems do not include capability/feature to 'build' application from source code
    - IaC focuses on infrastructure and not application

IaC vs PaaS

- IaC focuses on infrastructure provisioning
- Target User: Operations Engineer
- Uses IaaS behind the scene
- Can be used to deploy pre-built applications

- PaaS focuses on application deployment
- Target User: Application Developer
- Uses IaC/IaaS behind the scene
- Can be used to deploy pre-built as well as custom applications

What is enabled by PaaSes?

- Using PaaSes application developers can themselves do production application deployment!!
  - In old days (pre-cloud) this was not the case
    - Application developers developed application
    - Operations Engineers deployed applications

- PaaSes and IaC led to the emergence of DevOps

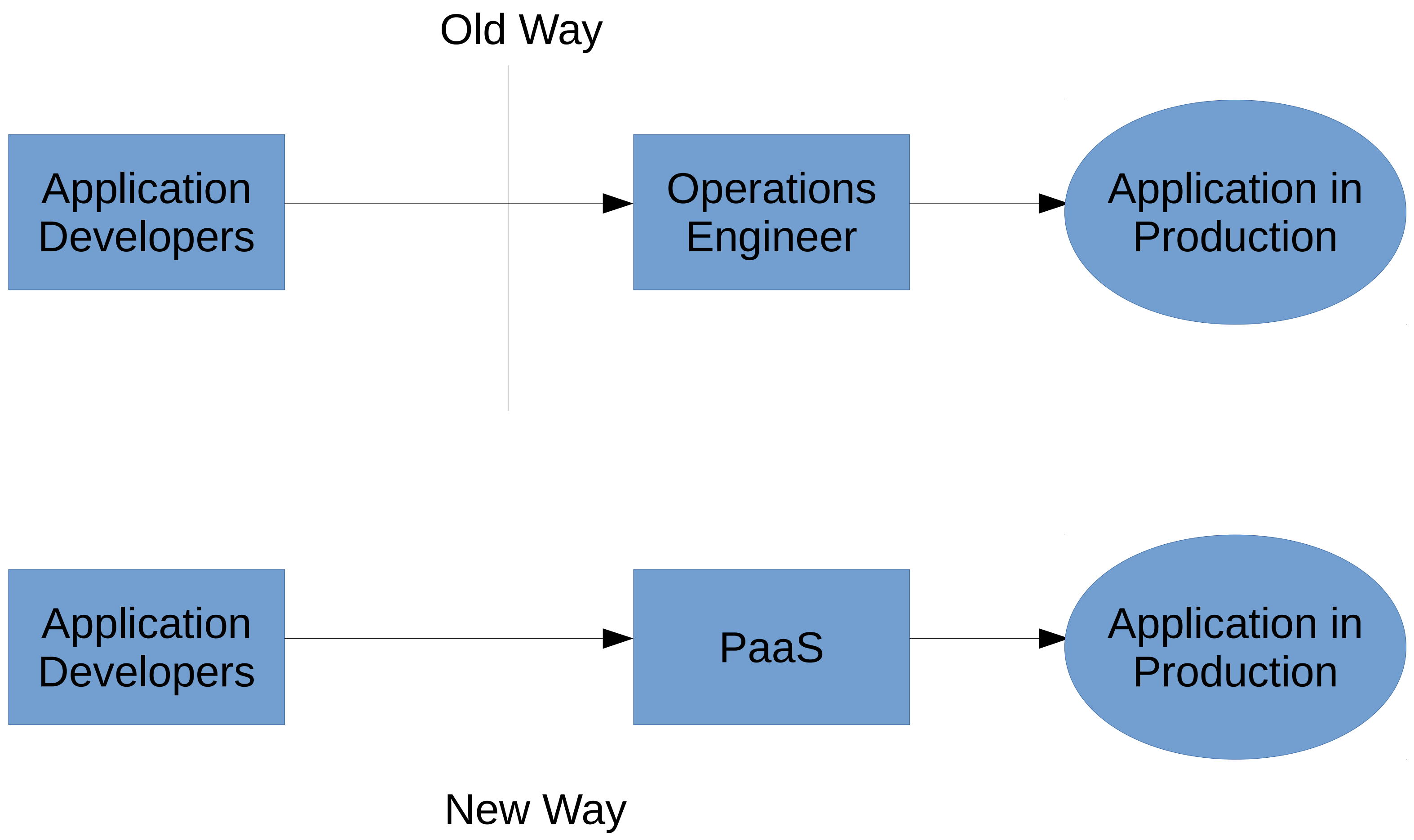
What is DevOps?

- Interpretation 1: It is a functional role
  - DevOps Engineer
- Interpretation 2: It is a set of steps/processes
  - DevOps process/practice
    - We will study this interpretation later

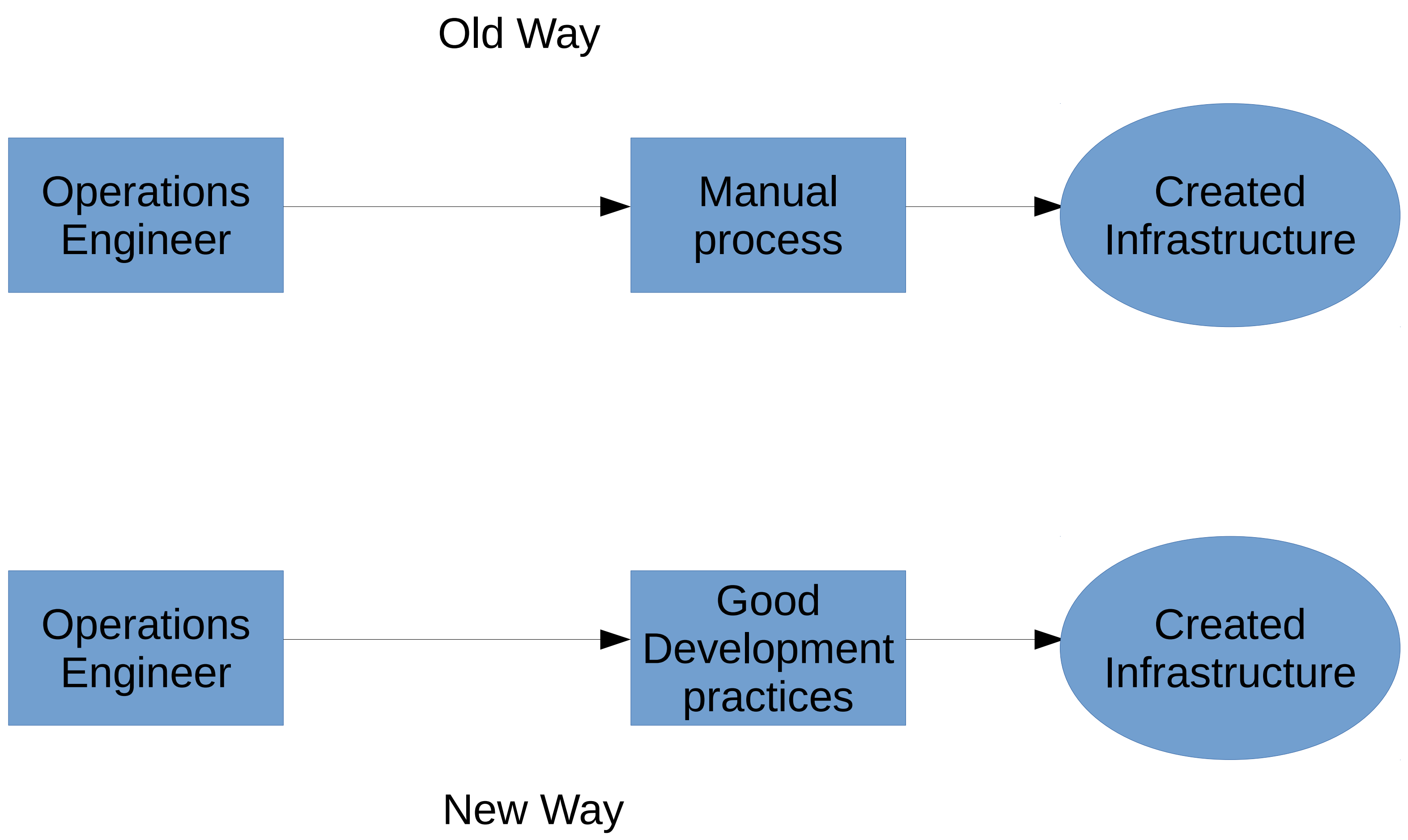
DevOps role: Who is DevOps?

- Who is DevOps?
  - Interpretation 1:
    - Developers who can (occasionally) perform Operations duties
  - Interpretation 2:
    - Operations Engineers who need to use “development practices” when setting up infrastructure
      - Codified infrastructure
      - Testing of infrastructure code
      - Version control of infrastructure code

DevOps role: Interpretation 1



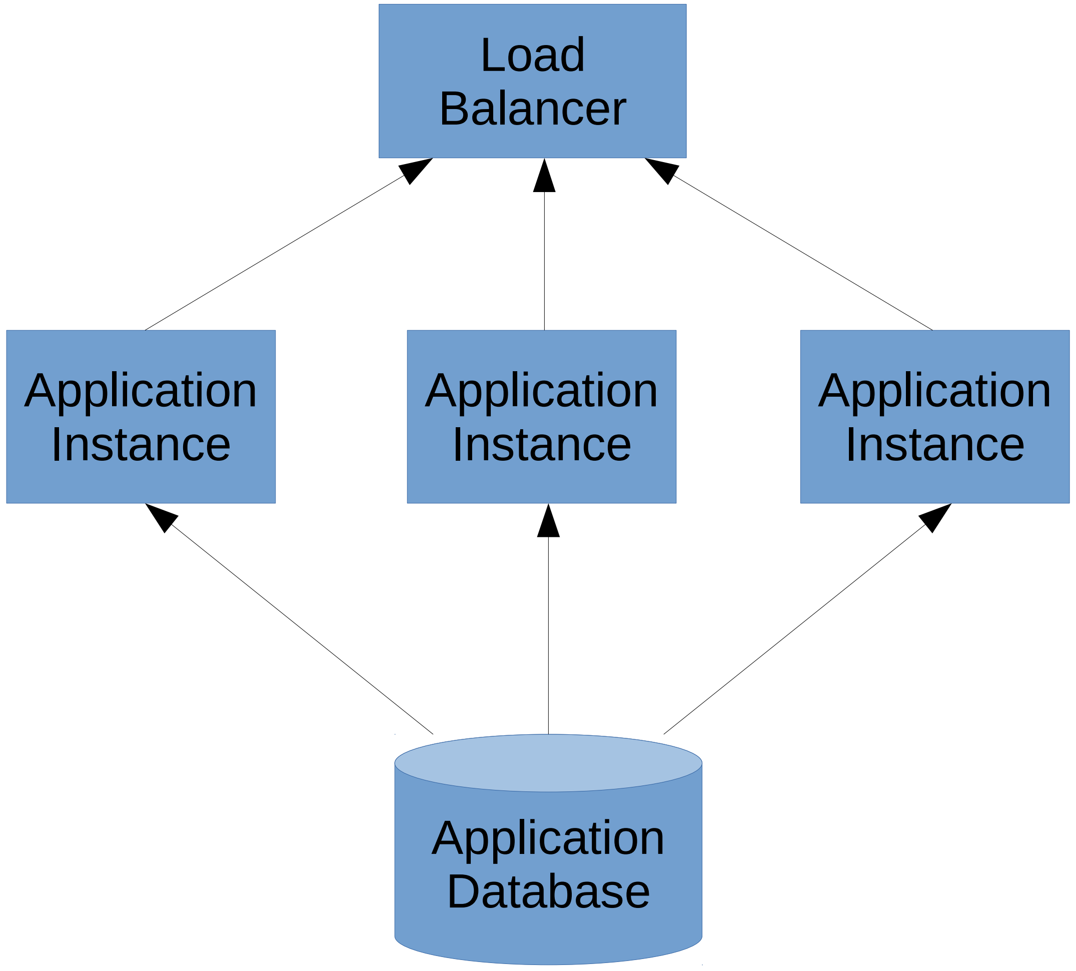
DevOps role: Interpretation 2



DevOps Role: Reality

- Application developers need to be aware of infrastructure
  - They should not treat infrastructure as black box
  - Beyond PaaSes, need to be aware of things such as IaC, configuration management, good security practices, etc.
    - CloudFormation, Terraform, Chef/Puppet, SSL
  - Actual work of infrastructure setup is still done by Operations Engineers
- Operations Engineers need to practice good development practices such as:
  - Writing unit and functional tests for their IaC templates
  - Following test-review-merge process for IaC templates
  - Closely work with application developers in defining application's production architecture and the required infrastructure

Web Application Deployment Architecture



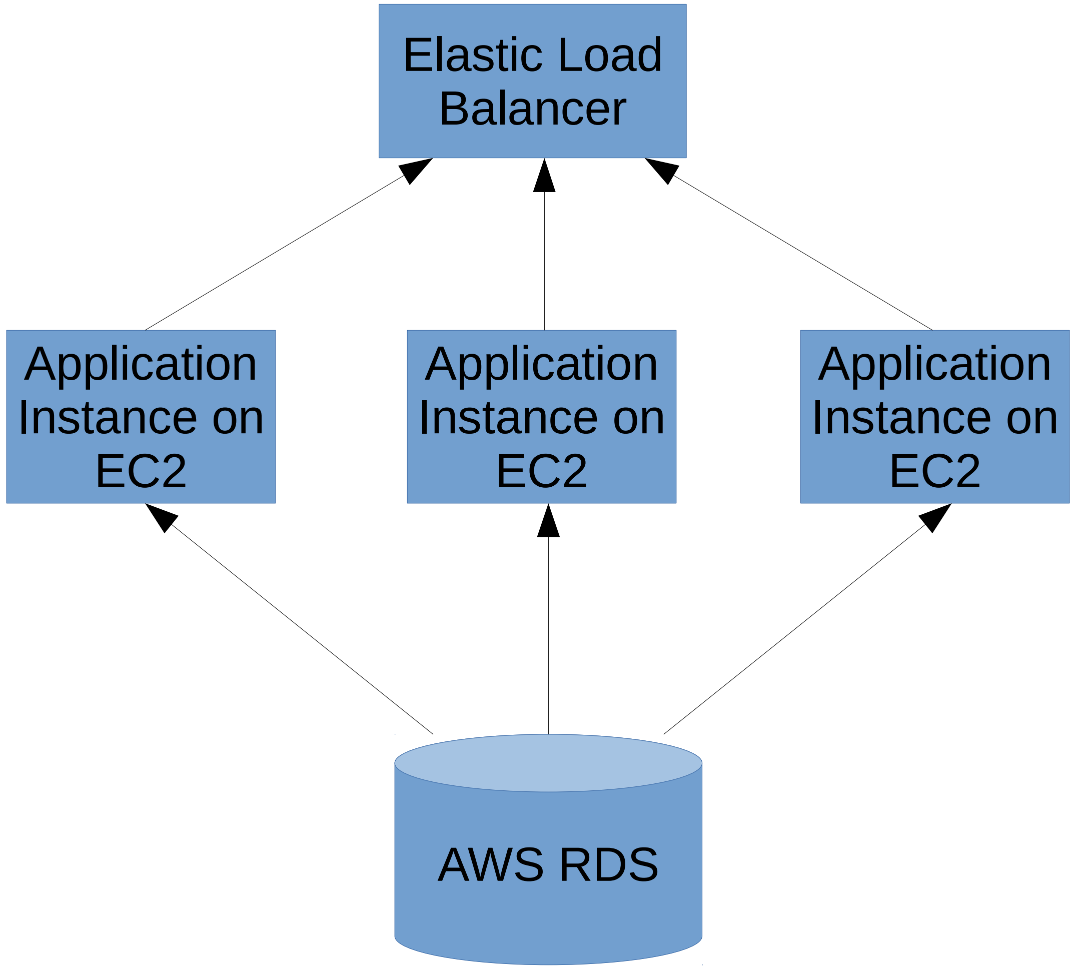
PaaS features

- Build and Deploy application
- Provide Load Balancer for application instances
- Provide DNS name for the application endpoint connected to the load balancer
- Provide ability to configure load balancer with SSL certificates
- Scale application instances
  - Manual scaling
  - Automatic scaling triggered by policies
- Test new application version with some subset of users
  - Canary deployment
- Perform application deployment without any downtime
  - Blue/green deployment
  - Rolling deployment
- Roll back deployed application to previous version
- Collect application logs
- Collect application metrics

AWS Elastic Beanstalk

- AWS PaaS
- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
- Supports applications developed in:
  - Java, PHP, .Net, Node.js, Python, Ruby

Web Application on AWS Beanstalk


Elastic Load Balancer
Application Instance on EC2
Application Instance on EC2
Application Instance on EC2
AWS RDS

Modern Web Applications

Devdatta Kulkarni

What are Web Applications?

Web pages vs. Web sites vs. Web Applications

What are Web Applications?

Web pages vs. Web sites vs. Web Applications

Web pages = static
Web applications = dynamic (+ more)
Web sites = Web pages + Web applications

Primary Characteristics of Web Applications

- Dynamic content
- Persistent storage
- API access (most of the time)
  - API = Application Programming Interface

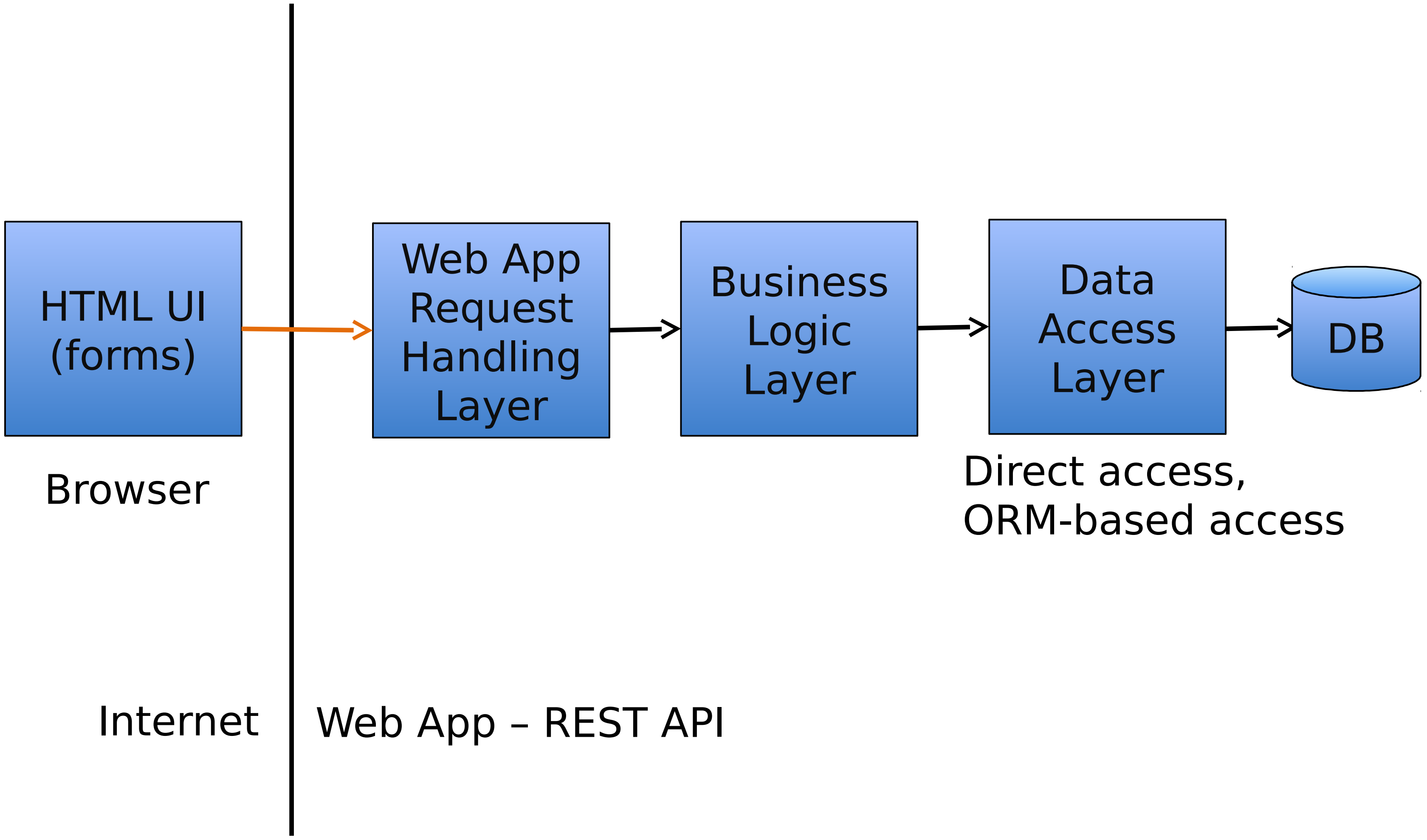
Other characteristics

- Robust and reliable
- Scalable
- Secure
- Responsive

Web Applications Elements

- Hypertext transfer protocol (HTTP)
- Framewors
  - Java Servlets,
  - Python Flask, Python Django
- Architecture
  - Logically divided into three layers

Modern Web Applications



AWS Elastic Beanstalk

Devdatta Kulkarni

Purpose of cloud resources

- Compute
- Storage
- Networks

Applications

Run applications

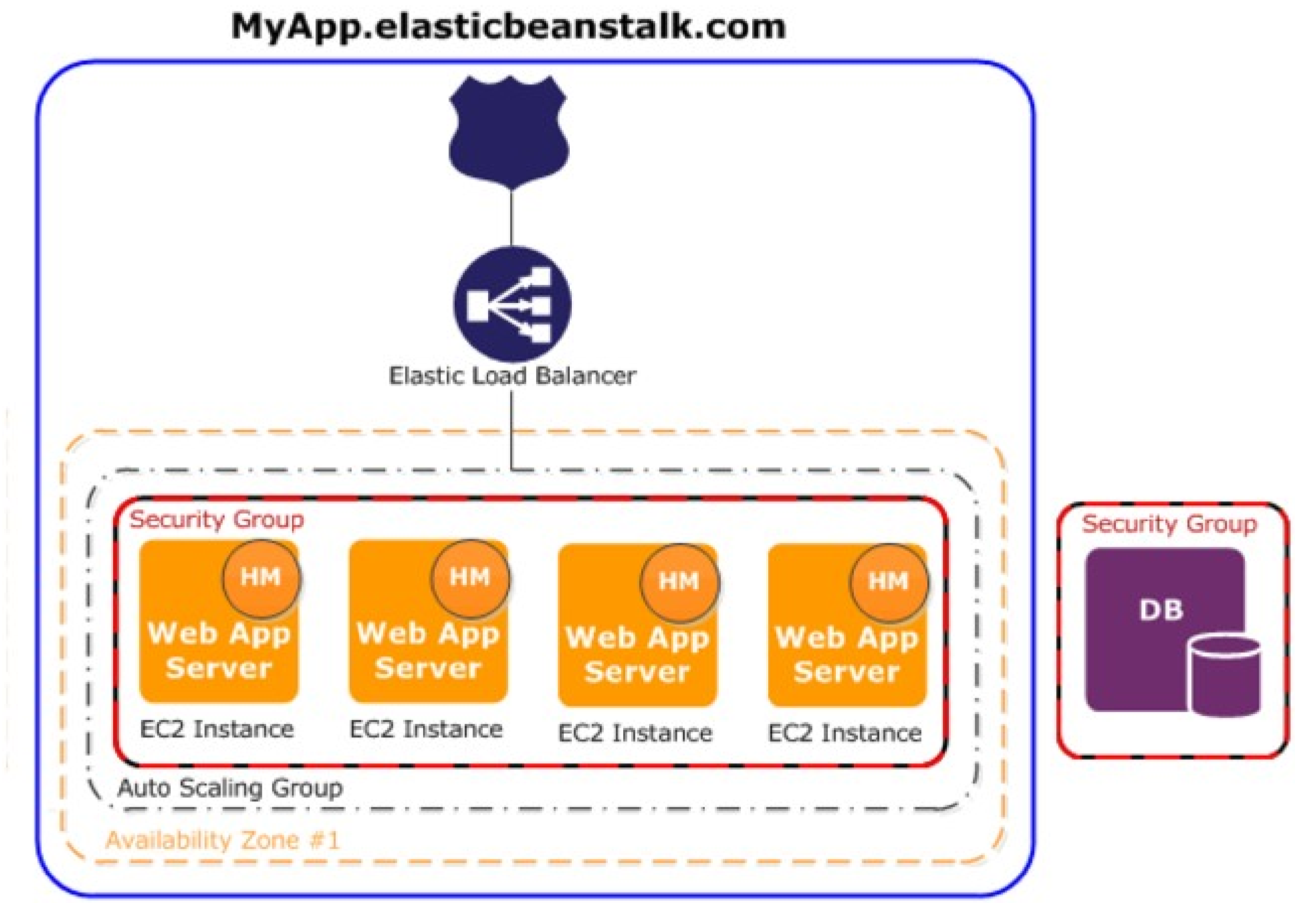
What is AWS Elastic Beanstalk?

- Amazon's Platform-as-a-Service (PaaS)
- It supports deployment of web applications starting from the source code
- Applications are deployed on EC2 instances
- Beanstalk handles provisioning of EC2 VMs and deploying applications on them
- Applications can use other AWS resources as well
  - RDS - Relational Database service
  - DynamoDB
  - S3
  - Etc.
- Beanstalk also provisions a Elastic LoadBalancer for the deployed application can provisions a publicly routable DNS record for it

Concepts

- Application
  - The OS folder/directory that contains the application's source code
- Application Version
  - Specific version of the application code
  - Application versions stored in S3
  - An application can have many versions
- Environment
  - An environment is application version that is *deployed* on a set of AWS resources
  - Each environment runs only a single application version
  - Beanstalk provisions the resources needed to run the application version
- Environment Tier
  - A environment tier determines whether the resources provisioned by Beanstalk would be used for deploying a web application that handles HTTP traffic or for deploying a back-end job that pulls tasks from a queue and works on them
- Environment Configuration
  - Environment configuration determines the parameters and the settings of environment's resources

Environment



Environment provisioning includes creation of:
- One instance of Elastic Load Balancer
- One instance of an Auto Scaling group
- One or more EC2 instances
- Creation of a CNAME record in Amazon Route 53 DNS web service
- The Auto scaling group adds/removes EC2 instances as and when needed

Host Manager

- Beanstalk software component that runs on each EC2 instance within an Environment
- Duties include
  - Deploying the application
  - Aggregating events and metrics for retrieval via the console, the API, or the command line
  - Generating instance-level events
  - Monitoring the application log files for critical errors
  - Monitoring the application server
  - Patching instance components
  - Rotating your application's log files and publishing them to Amazon S3

Environment Types

- Load-Balancing, Autoscaling Environment Type
  - Beanstalk will autoscale the number of EC2 instances that run your application
- Single-Instance Environment Type

IAM Permissions

- User Policies
  - Permissions that your IAM user needs to interact with Elastic Beanstalk
- Service Role
  - Needed by the Elastic Beanstalk service to interact with other AWS services on your behalf
- Instance Profile
  - The role and permissions that would be granted to each EC2 instance within your environment

User Policy

- Elastic Beanstalk will use the permissions given to the IAM User when creating other AWS resources

- The IAM user needs to be granted the AWSElasticBeanstalkFullAccess managed policy
  - This is pre-defined by AWS

User Policy

- AWSElasticBeanstalkFullAccess
- Elastic Beanstalk uses user permissions to *launch* all the resources in an Environment
  - Provision EC2 instances, Elastic Load Balancer, Auto Scaling group
  - Save logs and configuration templates to S3
  - Send notifications to Amazon SNS
  - Assign instance profiles to EC2 instances
  - Publish metrics to CloudWatch
  - Perform CloudFormation orchestration for resource deployments and updates
  - Amazon RDS permissions to create databases when needed
  - Amazon SQS permissions to create queues

Service Role

- Contains all the necessary permissions that allows Beanstalk to call other AWS services *pro-actively* (without user intervention)
  - Permissions that allow Elastic Beanstalk to monitor instance and environment health
    - AWSElasticBeanstalkEnhancedHealth Managed Service Role Policy
  - Permissions that allow Elastic Beantalk to update environments to perform managed updates
    - AWSElasticBeanstalkService Managed Service Role Policy
- aws-elasticbeanstalk-service-role created by default when creating an environment
  - 'eb create' command
  - Trusted Principal of this Role
    - elasticbeanstalk.amazonaws.com

User Policy vs. Service Role

- User Policy is granted to IAM User
- Elastic Beanstalk uses permissions defined through the User Policy when handling resource provisioning as part of *user-initiated Beanstalk actions (e.g.: 'eb create' command)*

- Service Role is granted to Elastic Beanstalk
- Elastic Beanstalk uses permissions defined through the Service Role when *pro-actively performing actions on an environment's resources (e.g.: monitoring health)*

Instance Profile

- AWSElasticBeanstalkWebTier Managed Policy
  - Policy statements that allow EC2 instances in your environment to upload logs to Amazon S3 and send debugging information to X-Ray and CloudWatch
- Default Instance Profile
  - aws-elasticbeanstalk-ec2-role
  - The Managed policy is attached to this default instance profile

AWSElasticBeanstalkWebTier Managed Policy

```
  {
 "Version": "2012-10-17",
 "Statement": [
  {
    "Sid": "BucketAccess",
    "Action": [
      "s3:Get*",
      "s3:List*",
      "s3:PutObject"
    ],
    "Effect": "Allow",
    "Resource": [
      "arn:aws:s3:::elasticbeanstalk-*",
      "arn:aws:s3:::elasticbeanstalk-*/*"
    ]
  },
  {
    "Sid": "XRayAccess",
    "Action": [
      "xray:PutTraceSegments",
      "xray:PutTelemetryRecords"
    ],
    "Effect": "Allow",
    "Resource": "*"
  },
  {
    "Sid": "CloudWatchLogsAccess",
    "Action": [
      "logs:PutLogEvents",
      "logs:CreateLogStream",
      "logs:DescribeLogStreams",
      "logs:DescribeLogGroups"
    ],
    "Effect": "Allow",
    "Resource": [
      "arn:aws:logs:*:*:log-group:/aws/elasticbeanstalk*"
    ]
  }
 ]
}
```

Platform/Container type

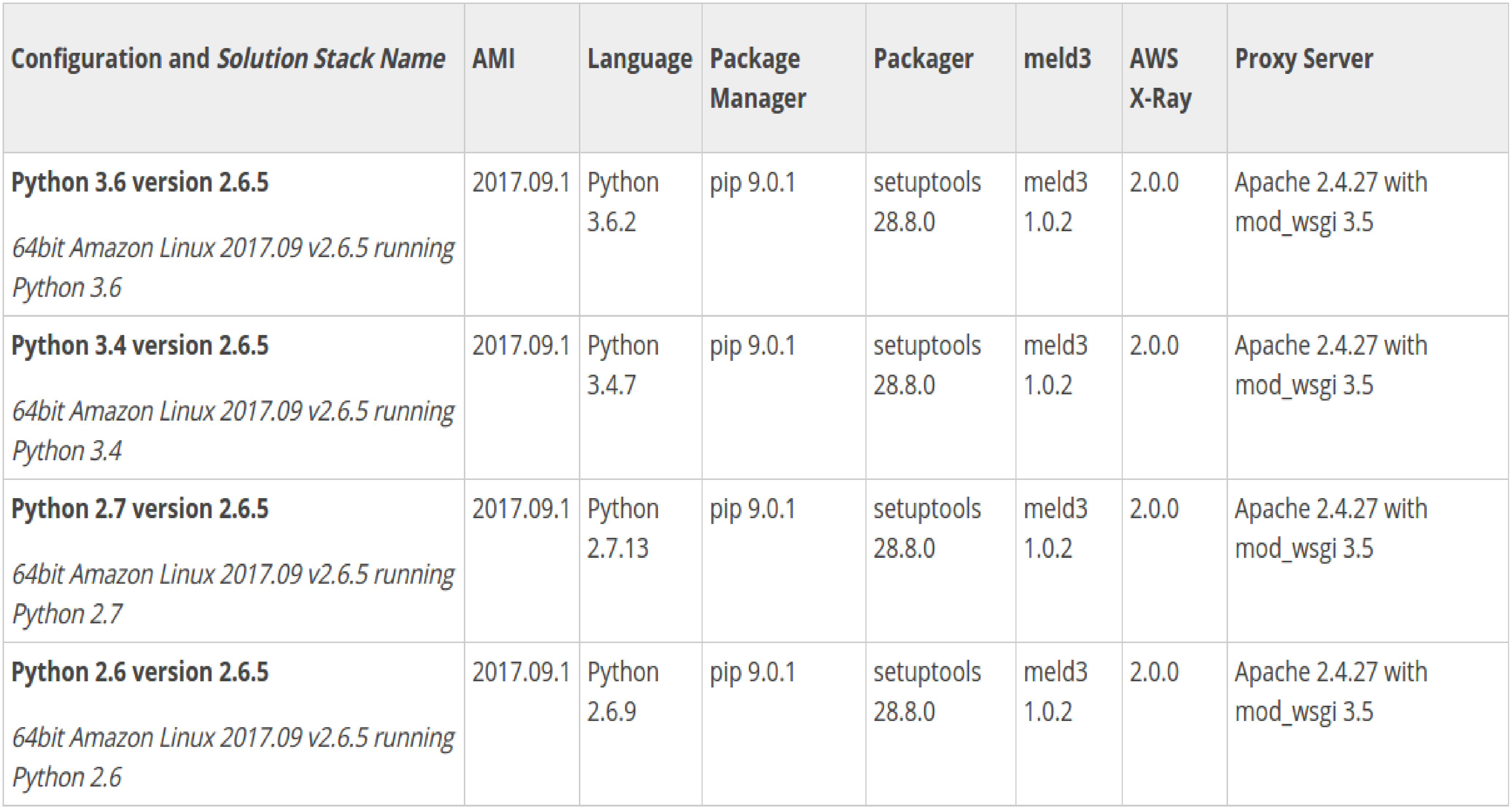
- What software stack to run on EC2 instances provisioned as part of an Environment?

- This is determined by the 'platform/container' type of the Environment
  - E.g.: Apache Tomcat container type for an Environment determines that the software stack that will be run on a EC2 instance will be: Apache web server and Apache Tomcat software running on Amazon Linux operating system
- The 'container' as used here is NOT same as the 'Docker' containers
  - These are application container platforms like Apache Tomcat

Platform Configurations

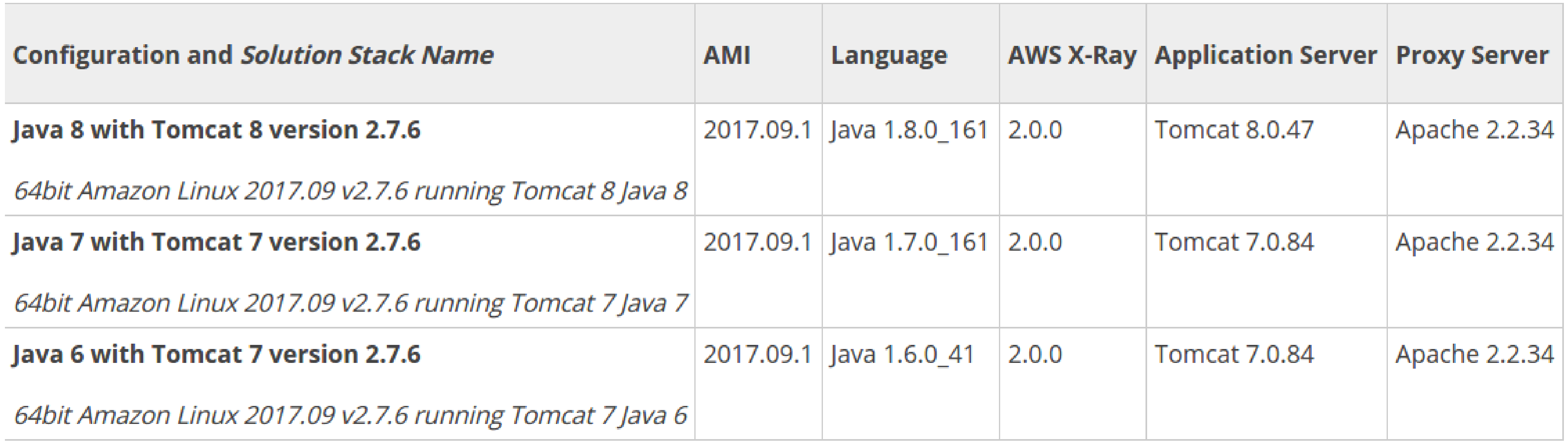
- Software stacks pre-defined by AWS
  - Go, Java SE, Java with Tomcat, .Net on Windows Server with IIS, Node.js, PHP, Python, Ruby
  - Single Container Docker
  - Multicontainer Docker
  - Preconfigured Docker
- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.platforms.html

Python Platform Configurations



Configuration and *Solution Stack Name*	AMI	Language	Package Manager	Packager	meld3	AWS X-Ray	Proxy Server
Python 3.6 version 2.6.5 *64bit Amazon Linux 2017.09 v2.6.5 running Python 3.6*	2017.09.1	Python 3.6.2	pip 9.0.1	setuptools 28.8.0	meld3 1.0.2	2.0.0	Apache 2.4.27 with mod_wsgi 3.5
Python 3.4 version 2.6.5 *64bit Amazon Linux 2017.09 v2.6.5 running Python 3.4*	2017.09.1	Python 3.4.7	pip 9.0.1	setuptools 28.8.0	meld3 1.0.2	2.0.0	Apache 2.4.27 with mod_wsgi 3.5
Python 2.7 version 2.6.5 *64bit Amazon Linux 2017.09 v2.6.5 running Python 2.7*	2017.09.1	Python 2.7.13	pip 9.0.1	setuptools 28.8.0	meld3 1.0.2	2.0.0	Apache 2.4.27 with mod_wsgi 3.5
Python 2.6 version 2.6.5 *64bit Amazon Linux 2017.09 v2.6.5 running Python 2.6*	2017.09.1	Python 2.6.9	pip 9.0.1	setuptools 28.8.0	meld3 1.0.2	2.0.0	Apache 2.4.27 with mod_wsgi 3.5

Java Platform Configurations



Configuration and *Solution Stack Name*	AMI	Language	AWS X-Ray	Application Server	Proxy Server
Java 8 with Tomcat 8 version 2.7.6 *64bit Amazon Linux 2017.09 v2.7.6 running Tomcat 8 Java 8*	2017.09.1	Java 1.8.0_161	2.0.0	Tomcat 8.0.47	Apache 2.2.34
Java 7 with Tomcat 7 version 2.7.6 *64bit Amazon Linux 2017.09 v2.7.6 running Tomcat 7 Java 7*	2017.09.1	Java 1.7.0_161	2.0.0	Tomcat 7.0.84	Apache 2.2.34
Java 6 with Tomcat 7 version 2.7.6 *64bit Amazon Linux 2017.09 v2.7.6 running Tomcat 7 Java 6*	2017.09.1	Java 1.6.0_41	2.0.0	Tomcat 7.0.84	Apache 2.2.34

PaaS Criticisms

- Too Opinionated
  - This means that it is not possible to change the software stack used by your application
    - This is now changing with
      - Docker-based container solutions
      - Support for “Custom Platforms” in Elastic Beanstalk
- Less Control
  - This means it is not possible to control configuration and setting of underlying IaaS resources
    - Most PaaS systems fall into this category
    - But Beanstalk allows IaaS resource control using a innovative approach!
- Cloud Vendor lock-in
  - Typically a web application deployed on one PaaS cannot be easily migrated to other PaaSes
    - Changing with PaaS agnostic tools like FirstMile
- Costly and difficult to use when applications are still being developed
  - Tools like FirstMile are aimed towards solving this problem
    - Note that just Docker is not enough – applications may depend on managed services like RDS instances. It should be possible to provision and use such services during application development. Docker does not help with this

Customizing Environment in Beanstalk

- All AWS resources provisioned as part of an Environment can be customized
  - EC2
  - Load Balancer
  - RDS
  - Autoscaling groups
- Include a configuration file within your application folder

Configuration Files for customization

- What things can be customized?
  - Software stack settings on EC2 instances
    - Additional Operating System packages that your application depends on
    - Different settings and configuration file
  - Flavor and instance type of provisioned resources

- The .ebextensions directory
  - Create this directory in the top-level folder of your application
  - Add files with “.config” extension to it

- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html

.ebextensions config file options

```
option_settings:
  - namespace:  aws:elasticbeanstalk:container:tomcat:jvmoptions
    option_name:  Xmx
    value:  256m
  - option_name: MYPARAMETER
    value: parametervalue
```

The option_settings values are injected as environment variables on the EC2 instance

.ebextensions config file options

```
packages:
  yum:
    libmemcached: []
    ruby-devel: []
    gcc: []
  rpm:
    epel: http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm
  rubygems:
    chef: '0.10.2'
```

Packages that should be downloaded on the EC2 instance

.ebextensions config file options

```
groups:
  groupOne: {}
  groupTwo:
    gid: "45"
```

Operating Systems 'groups' that should be created on the EC2 instance

.ebextensions config file options

```
users:
  myuser:
    groups:
      - group1
      - group2
    uid: "50"
    homeDir: "/tmp"
```

Operating System users that should be created on the EC2 instance

.ebextensions config file options

```
sources:
  /etc/myapp:
https://s3.amazonaws.com/mybucket/myobject
```

Download and unpack a archieve at the specified path on the EC2 instance

.ebextensions config file options

```
files:
  "/home/ec2-user/myfile" :
    mode: "000755"
    owner: root
    group: root
    source: http://foo.bar/myfile

  "/home/ec2-user/myfile2" :
    mode: "000755"
    owner: root
    group: root
    content: |
      # this is my file
      # with content
```

Create specified file on the EC2 instance

.ebextensions config file options

```
commands:
  command name:
    command: command to run
    cwd: working directory
    env:
      variable name: variable value
    test: conditions for command
    ignoreErrors: true
```

Commands are run before the application and web server are set up and the application version file is extracted

.ebextensions config file options

Services:

Use services section to define the services/processes that you want started on your EC2 instances

Container commands:

These are the commands are run after the application and web server have been set up and the application version archive has been extracted

.ebextensions for AWS Resources

- Use CloudFormation snippets
  - To customize AWS resources

- Unique to AWS Beanstalk!!

.ebextensions for AWS Resources

```
Resources:
  SessionTable:
    Type: AWS::DynamoDB::Table
    Properties:
      KeySchema:
        HashKeyElement:
          AttributeName:
            Fn::GetOptionSetting:
              OptionName : SessionHashKeyName
              DefaultValue: "username"
          AttributeType:
            Fn::GetOptionSetting:
              OptionName : SessionHashKeyType
              DefaultValue: "S"
      ProvisionedThroughput:
        ReadCapacityUnits:
          Fn::GetOptionSetting:
            OptionName : SessionReadCapacityUnits
            DefaultValue: 1
        WriteCapacityUnits:
          Fn::GetOptionSetting:
            OptionName : SessionWriteCapacityUnits
            DefaultValue: 1
```

Relational Database Service (RDS)

- Beanstalk can create a RDS instance as part of creating an environment
  - Life-cycle of such a RDS instance is tied with that of the environment
  - When environment is terminated, the RDS instance will terminate
- Beanstalk provides connection information to your application by setting environment properties for database hostname, port, user name, password, db name
  - RDS_HOSTNAME
  - RDS_PORT
  - RDS_DB_NAME
  - RDS_USERNAME
  - RDS_PASSWORD
- In your application code you need to read values for these variables

Persistent Storage using Databases

Devdatta kulkarni

What is Persistent Storage?

- Storage medium that supports storage of data independent of whether an application program is running or not
  - Examples of persistent storage
    - Files written to disk
    - Records written to a database (topic of today’s lecture)
- Opposite of persistent storage
  - In-memory storage (Main memory, random access memory, memory)
  - Examples of Non-persistent storage
    - Objects in our Java applications
      - The object “exists” only while the program is running

What is a database?

- Persistent storage in which data is in one or more *tables*
- Data is managed by database server
- A table has one or more *columns*

Retrieving data from a database

- How to retrieve data from a database table?
  - Using Structured Query Language (SQL)

- What is SQL?
  - SQL is a *declarative language* that supports writing queries over a set of tables to retrieve records that match the specified query criteria

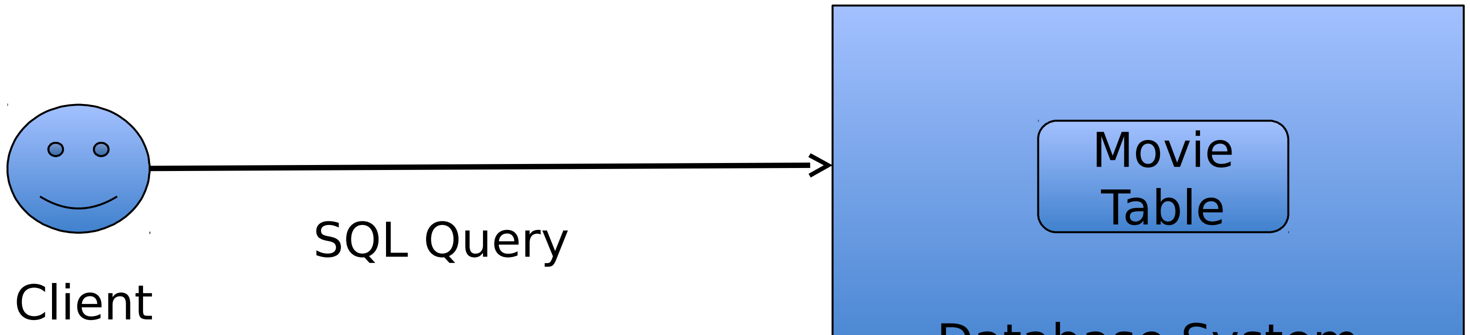
SQL

- What is a declarative language?
  - It is a language in which you write code identifying “what” the answer should look like instead of writing code identifying “how” to produce the answer

- Suppose we wanted to retrieve records of all the students living in Austin from the Student table:
  - In the declarative model, we will write a query of the following form and send it to the database system which manages all the database tables

“Retrieve all records from the Student table for which City column is equal to Austin”

Querying a Database system



SQL Statements

- SQL Insert == Create
- SQL Update == Update
- SQL Delete == Delete
- SQL Select == Read

SQL Examples

- http://www.w3schools.com/sql/default.asp

- Create
  - http://www.w3schools.com/sql/sql_insert.asp
- Read
  - http://www.w3schools.com/sql/sql_select.asp
- Update
  - http://www.w3schools.com/sql/sql_update.asp
- Delete
  - http://www.w3schools.com/sql/sql_delete.asp

SQL

- Some of the most important SQL commands
  - http://www.w3schools.com/sql/sql_syntax.asp
    - Create database
    - Create table
    - Select
    - Update
    - Delete
    - Insert Into
    - Alter database
    - Alter table
    - Drop table

Installing MySQL

- MySQL Community Server
  - http://dev.mysql.com/downloads/
    - On Mac, download the .dmg file
- Starting MySQL Server (on Mac)
  - In Spotlight search for MySQL
  - MySQL window should open up with “Start MySQL Server” button on it
  - MySQL Server Instance should be in “stopped” status
  - Hit the “Start MySQL Server” button; you may be prompted for a password
  - Enter the password; MySQL server should be up

Installing MySQL

- MySQL Clients
  - Command line
    - /usr/local/mysql/bin/mysql – if you install the .dmg from above link
  - MySQL Workbench
  - SQuirrel SQL
    - http://squirrel-sql.sourceforge.net/
- Local MySQL Server
  - 127.0.0.1
  - Port number 3306
  - Usename: root

Setting up the Database

- Run mysql client
  - /usr/local/mysql/bin/mysql -u root
- Create Database
  - create database movies_db;
- Create User
  - Example: create user 'devdatta'@'localhost’;
- Grant Privileges
  - GRANT ALL ON movies_db.* TO ‘devdatta’@’localhost’;
- Logout
  - On mysql prompt: “quit”
- Login as user;
  - /usr/local/mysql/bin/mysql –u devdatta –h localhost
- Create Table
  - use movies;

Problem

- Design a database schema to represent the following:
  - Courses in the CS department
  - Students in the CS department
  - A student can take at most one course
  - A course can be taken by zero or more students
- We want to support following queries:
  - Find all the students who are taking a particular course
  - Find all students and the courses that they are taking

Designing DB

- A *course* table to represent courses
- A *students* table to represent students
- A *relationship* from the *students* table to *courses* table to indicate that a student is taking a particular course

Courses table

A row in the table represents one course

Table columns:

- Name of the course
- Course number
- A unique id

Constraints:

- Name: Should not be NULL
- Course number: Should not be NULL
- Unique id: This will be the PRIMARY KEY of the table

Primary key

- What is a primary key?
  - Any attribute that can uniquely identify a row in the table

- If the resource has a unique attribute, can we use that as the primary key (natural key)?
  - Examples:
    - SSN number to represent a person
    - UTEID to represent a student
- Or, should we generate unique ids and use those (surrogate key)

Courses table

```
create table courses(name varchar(255)
NOT NULL, course_num varchar(20) NOT
NULL, course_id int NOT NULL
AUTO_INCREMENT, PRIMARY
KEY(course_id));
```

course_id is the *surrogate key*

AUTO_INCREMENT:
- Let the database generate the values for course_id

Student table

A row in the table represents one student
A student can take at most one course

Table columns:

- Name of the student
- A unique id
- A column to capture the relationship between a student and a course

Constraints:

- name: Should not be NULL
- Unique id: This will be the PRIMARY KEY of the table
- course_id: A FOREIGN KEY to courses table

Foreign Key

- What is a Foreign key?
  - A column in a table that refers to primary_key column in another table

Student table

create table students(name varchar(255) NOT NULL, student_id int AUTO_INCREMENT, course_id int, PRIMARY KEY(student_id), FOREIGN KEY(course_id) REFERENCES courses(course_id));

student_id is the *surrogate key*

AUTO_INCREMENT:
- Let the database generate the values for student_id

Populate the tables

- insert into courses(name, course_num) values("Data Management", "CS347");

- insert into students(name) values("Student 2");

- insert into students(name, course_id) values("Student 4", (select course_id from courses where course_num="CS378"));

Queries

- Queries:
  - Find all the students who are taking a particular course
  - Find all students and the courses that they are taking

Join

- A *join* is a mechanism that allows writing of queries over several tables

- A join of two tables selects all the rows from both the tables as long as there is a match between the specified columns of the two tables

Types of Joins

- Inner Join
  - If we imagine tables being sets, then inner join can be thought of as *intersection* of the two sets with the intersection criteria being the matching column values on the join column
- Left outer join
  - Inner join + rows from the “left” table which may not have matched on the join column
- Right outer join
  - Inner join + rows from the “right” table which may not have matched on the join column
- Full outer Join
  - Left outer join + Right outer join – duplicates
    - MySQL does not support it

Join

- Find all students who are taking course CS378

```
select * from students join courses on
students.course_id =
courses.course_id where
courses.course_num="CS378";
```

Join

- Find the students who are taking at least one course and find the information of the course that they are taking

```
select * from students join courses on
students.course_id = courses.course_id;
```

Left outer join

- Find all the students irrespective of whether they are taking any course; for those who are taking at least one course, find the information about the course

```
select * from students left outer join courses
on students.course_id = courses.course_id;
```

Right outer join

- Find all the courses irrespective of whether a course is being taken by any student. For the course that is being taken by at least one student, find the information about all those students

```
select * from (students right outer join
courses on students.course_id =
courses.course_id);
```

References

- SQL Tutorial
  - http://www.w3schools.com/sql/
- Primary key vs. surrogate key
  - http://stackoverflow.com/questions/2186260/when-to-use-an-auto-incremented-primary-key-and-when-not-to
  - http://stackoverflow.com/questions/63090/surrogate-vs-natural-business-keys

Docker

Devdatta Kulkarni

Problem

- How to make application deployments *repeatable*?
  - What does this mean?
    - It should be possible to repeatedly deploy an application, always getting the same result
- Why is this an important problem?
  - Applications that are developed and test locally, should work exactly like that in production
    - To provide confidence in application development and testing process
- In general applications should behave same across different “environments”
  - Local development, testing environment, production environment, etc.

Repeatability Issues

- Issue 1: Similarity between environments
  - How to ensure that the environment components are similar across environments? For e.g., each environment has a database, a loadbalancer, given number of VMs, etc.

- This issue is typically solved by using PaaS or IaC for deployment
  - For instance, using Elastic Beanstalk if same application can be deployed in different environments.

Repeatability Issues

- Issue 2: Consistency of packages and libraries
  - How to ensure that the packages and libraries that are used to build the application are consistent across different environments?
    - Language libraries
    - OS packages

Issue - Consistency of packages and libraries

- Consistency of Language libraries is a solved problem
  - Python has requirements.txt and pip
  - Java has pom.xml and maven
  - Nodejs has package.json and npm
- What about OS-level packages?
  - Certain applications may run only on certain OSes
    - E.g.: FirstMile currently runs only on Ubuntu and Mac OS
- Idea:
  - Is it possible to somehow package application's Operating System along with the application?
    - Similar to requirements.txt, pom.xml, package.json
  - This will allow us to deploy applications without worrying about incompatible OS-level packages!!

Challenges in packaging OS along with application

1) How to define which OS we want?
   - Different applications may need different Operating systems

2) What would be the end result of this packaging?
   - In Java world when we package an application we get a ".war" file
   - In Python, we get the virtualenv directory
   - Similar to above, what would be the packaged format of App+OS look like?

Challenges in packaging OS along with application

3) Where will this App+OS package run?

- Can it run directly on base OS?
- Can it run within a VMs like EC2 instances?

4) How does an application bind to dependent services like databases?

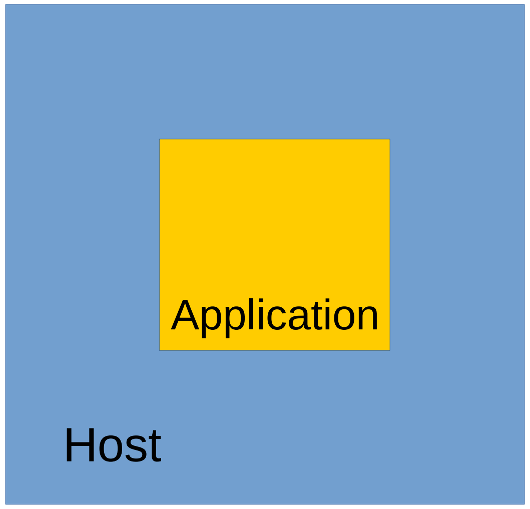
5) How will the application be accessible?

Problems

- Repeatability
  - We have already talked about it
- Utilization
  - Modern Hosts have large capacity
    - Containers allow us to maximize the utilization of a Host

Utilization

- What fraction of time is the Host busy doing relevant work?



Non-Container model

If Application is busy service requests for 15 minutes within an hour then the Utilization of the Host is:

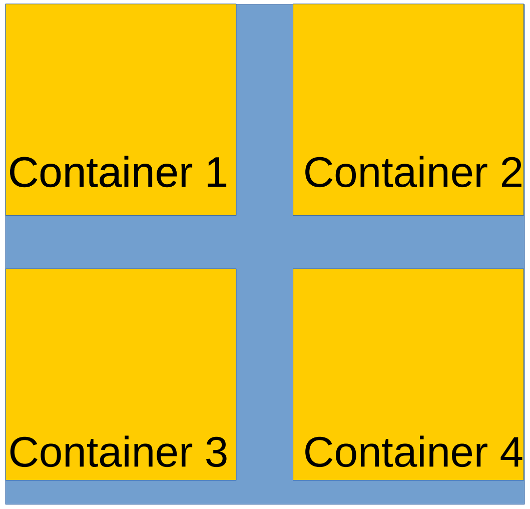
U = 15 minutes / 60 minutes

U = 0.25

U = 25%

Utilization

- What fraction of time is the Host busy doing relevant work?



Container-based model

If each Container receives traffic in different 15 minute period
Within an hour then the Utilization of the Host is:

U = 60 minutes / 60 minutes

U = 1

U = 100%

Problems

- Repeatability
  - We have already talked about it
- Utilization
  - Modern Hosts have large capacity
    - Containers allow us to maximize the utilization of a Host
- Resource Isolation

Resource Isolation

- This becomes an issue since we are packing more than one Container on a Host

- Isolation requirements:
  - Root file systems of Containers should remain separate
  - Processes that are running “inside” Containers should remain separate
  - Network traffic destined for different Containers should remain separate
  - A rogue container should not be able to affect other Containers running on the Host

What is Docker?

- Technology for packaging OS + Application
  - https://www.docker.com/

- History
  - Started as a technology differentiator within a PaaS called “DotCloud” – commercial product
  - Core Docker technology open sourced in 2013
    - DotCloud company got renamed to Docker Inc.

Docker concepts

- Dockerfile
- Base Image
- Running Docker container
- Ports
- Environment variables
- Volumes
- Networking

Dockerfile

- Declarative definition format
- Allows defining:
  - Base Operating System
  - Any additional packages that we want to install on it
  - Where to inject application within the OS image?
  - What environment variables to set?
  - What port the application will listen on?
  - Etc.

Dockerfile Example

```
FROM ubuntu:14.04
RUN apt-get update -y \
    && apt-get install -y python-setuptools python-pip
ADD requirements.txt /src/requirements.txt
RUN cd /src; pip install -r requirements.txt
ADD . /src
EXPOSE 5000
CMD ["python", "/src/application.py"]
```

Dockerfile - Example

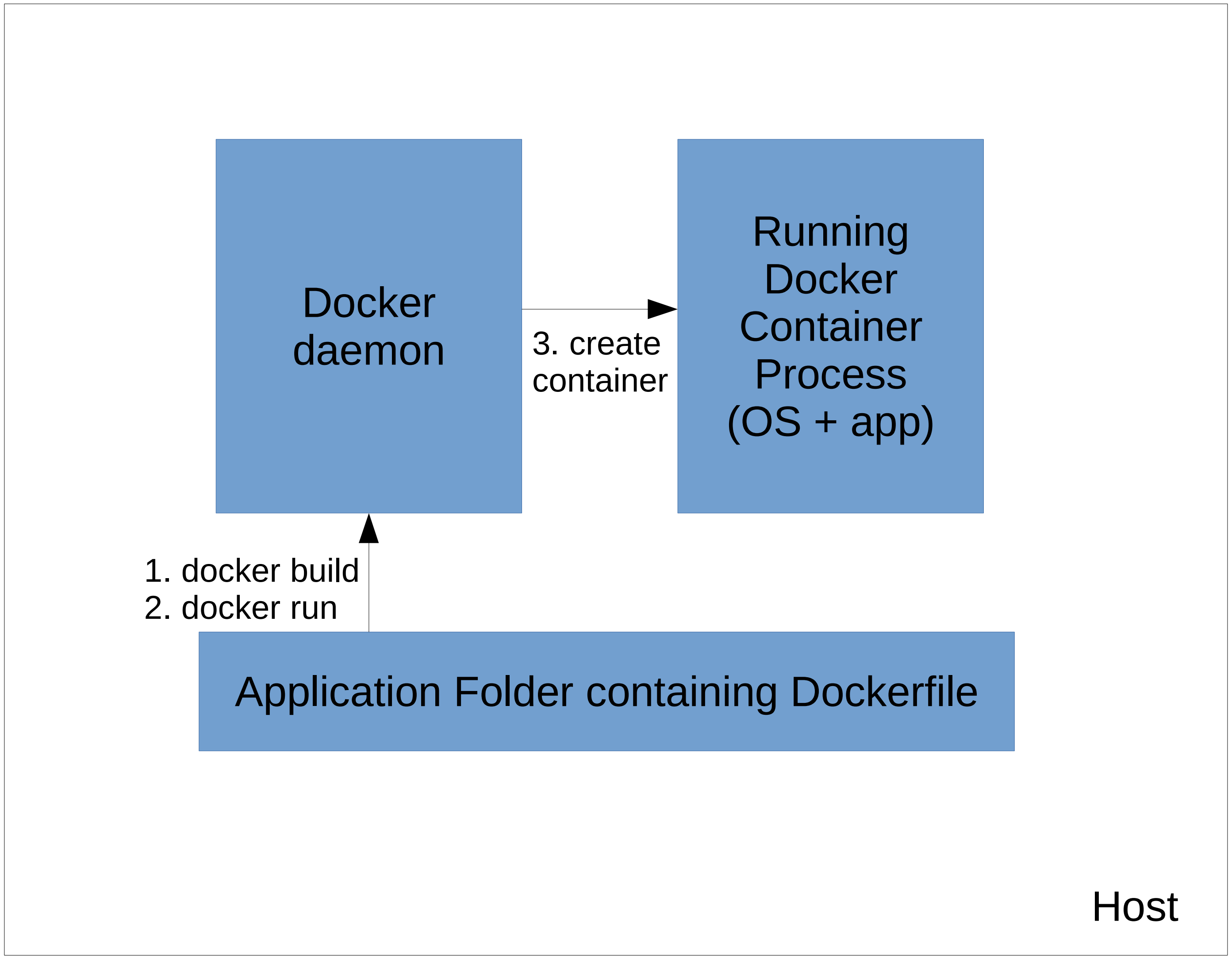
```
FROM ubuntu:14.04
RUN apt-get update -y \
    && apt-get install -y python-setuptools python-pip
ADD requirements.txt /src/requirements.txt
RUN cd /src; pip install -r requirements.txt
ADD . /src
EXPOSE 5000
ENV  PASSWORD testpass123!#$
ENV  DB testdb
ENV  HOST 172.17.0.7
ENV  USER testuser
CMD ["python", "/src/application.py"]
```

Dockerfile details

- Default name: Dockerfile

- Location:
  - Dockerfile should be put at the top location in the application folder

How does Docker work?


Docker daemon
3. create container
Running Docker Container Process (OS + app)
1. docker build
2. docker run
Application Folder containing Dockerfile
Host

How does Docker work?

- Docker daemon pulls down the base OS image that is specified in the Dockerfile

- Docker daemon then executes each line defined in the Dockerfile on this base image

- Docker daemon builds the final image in the form of “layers”

- Every line in the Dockerfile causes a new “diff layer” to be generated over the previous layer

How are all the challenges addressed?

Issue	**Solution**
How to define which OS we want?	Dockerfile → FROM command
End result of Packaging	Docker image
Where will Docker Image run?	Any Host (physical machine, VM)
How does application bind to dependent services?	Dockerfile → ENV command
How will the application be accessible?	Custom application URLs (through host ports)

Basic Docker Commands

- Build a Docker container
  - docker build -t <tag-name> .
    - Dockerfile needs to be present in the directory from which this command is executed
- Execute a Docker container
  - docker run -d -p <host-port>:<container-port> <tag-name>

    -d: Run the Docker container in detached mode

    -p: Map container-port to host-port

    <tag-name>: Name used in 'docker build' command
- See all local Docker images
  - docker images

Basic Docker Commands

- See logs of running Docker container
  - docker logs <container-id>
- See all running Docker containers
  - docker ps
- Stop a running Docker container
  - docker stop <container-id>
- Remove a stopped Docker container
  - docker rm <container-id>
- Remove a Docker container image
  - docker rmi <tag-name>

Differences between VMs and Containers

- A VM image is a full OS image
- Building a VM image takes time (10s of minutes)
- Starting up a VM instance from VM image takes time (several minutes)
- VMs runs on a Hypervisor
- No one standardized declarative format for constructing a VM image
  - Disk-Image-Builder is one such somewhat popular tool

- Container images are created from base OS image as layers
- Building Container image is relatively faster (sometimes minutes)
- Starting Containers is very fast
- Containers run as a Process on a Host
- Dockerfile provides a standardized declarative format for constructing a Docker container image

Design Principles of Modern Web Applications

The Twelve-Factor App Manifesto

– https://12factor.net/

- Developed by Heroku

- Docker and Container technology satisfy several of the 12factor principles

Twelve Factor App Manifesto

1) Codebase tracked in version control
   - Solved using Git, etc.
2) Explicitly declare and isolate dependencies
   - Solved using requirements.txt, pom.xml, package.json
3) Store application Config in the environment
   - Solved using ENV command in Dockerfile
4) Treat backing services as attached resource
   - Solved using ENV command in Dockerfile

Twelve Factor App Manifesto

5) Separate build and run stages

- Achieved through separation of 'docker build' and 'docker run' commands

6) Execute application as Processes

- Docker containers run as Processes on Host OS

7) Expose applications through Ports

- Achieved through:
    - EXPOSE command in Dockerfile and “-p <host-port>:<container-port>” flag in 'docker run'

8) Scale out application via the process model

- Achieved through running more than one Docker containers (each running container needs to be bound to different host port)

Twelve-Factor App Manifesto

9) Fast application startup and graceful shutdown
   – Docker containers are very fast to start up

10) Parity between environment (Dev/test/prod)
   – Docker container image once built can be run on any Host

11) Generate application log streams
   – Achieved through 'docker logs' command

12) Run admin tasks as one-off processes
   – Can be achieved through running admin Docker containers

Container Orchestration

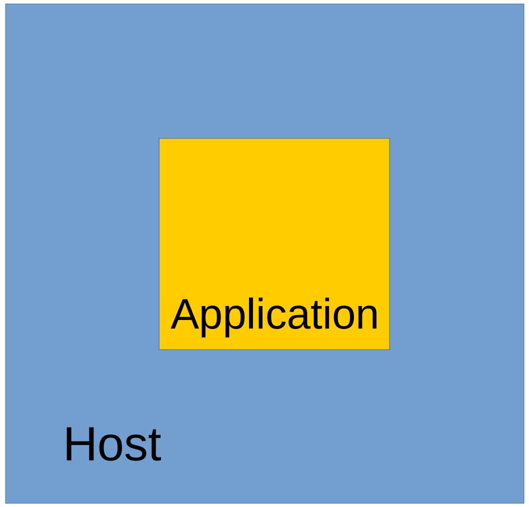
Devdatta Kulkarni

Advantage of Containers

- Utilization of Host
- Repeatability of deployments
- Abstract away the Host Operating System from Applications
  - A Docker container image contains Base OS needed by the application – the Host OS stops mattering to the application!!

Utilization

- What fraction of time is the Host busy doing relevant work?



Non-Container model

If Application is busy service requests for 15 minutes within an hour then the Utilization of the Host is:

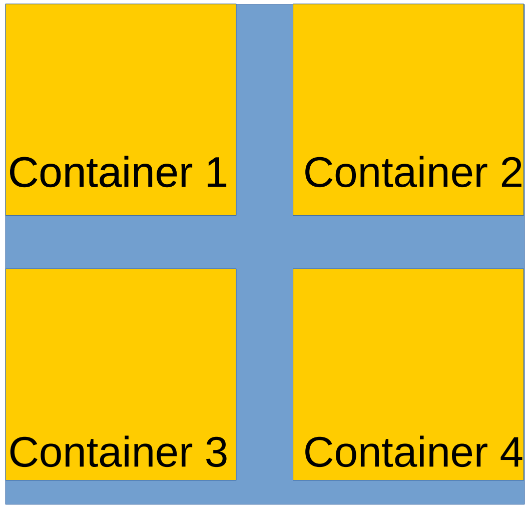
U = 15 minutes / 60 minutes

U = 0.25

U = 25%

Utilization

- What fraction of time is the Host busy doing relevant work?



Container-based model

If each Container receives traffic in different 15 minute period
Within an hour then the Utilization of the Host is:

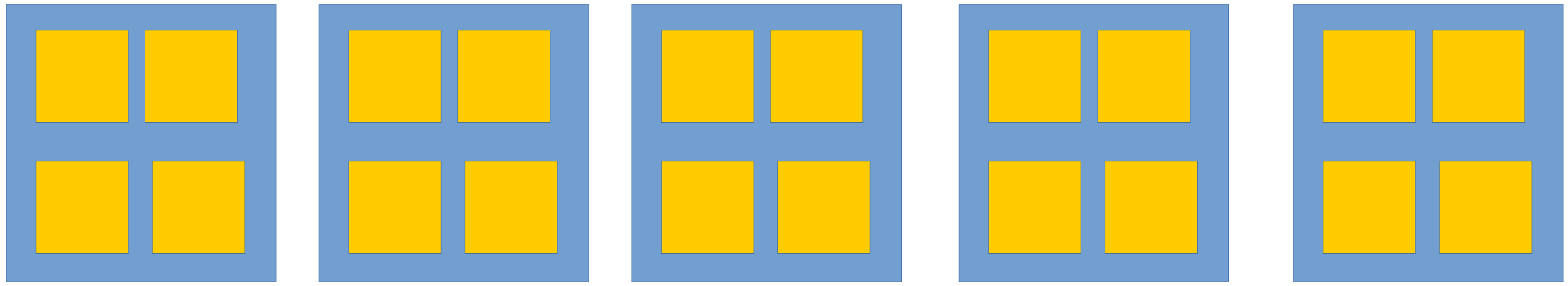
U = 60 minutes / 60 minutes

U = 1

U = 100%

Problem

- Container Orchestration
  - How to run containers across number of hosts?



Container Orchestration Systems

- Docker Swarm
- Apache Mesos
- Amazon Elastic Container Service (ECS)
- Kubernetes

- Kubernetes is the most advanced amongst these systems
  - Also it is becoming most popular/used

Borg, Omega, Kubernetes

- Omega:
  - https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41684.pdf
- Borg, Omega, Kubernetes:
  - https://research.google.com/pubs/archive/44843.pdf

Omega

- Global cluster state
  - Shared persistent store
- Multiple schedulers
  - Each scheduler maintains a local copy of the Global state
- Optimistic concurrency control
  - Each scheduler can make scheduling decisions independently of others
  - If resources are being used by some other task, they are preempted and given to the requesting scheduler depending on Job Priority

Lessons Learned through Borg, Omega, Kubernetes

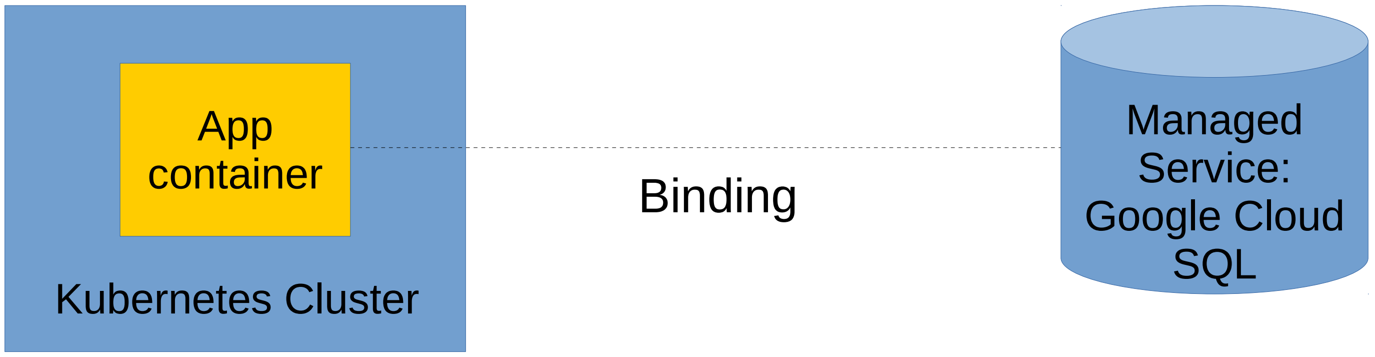
- Allocate separate IP address for each running container
  - Don't try to separate container URLs solely based on port numbers that are allocated by the Container Orchestration Engine (COE)
    - Borg used to do this
- Use a higher-level grouping mechanism for grouping containers
  - When containers corresponding to one application are running on several hosts, we need a way to address them together
  - Kubernetes uses the mechanism of *Labels* and *Label selectors* for this purpose
    - Containers can be assigned Labels
      - Search/Delete/Update queries can then be written based on Labels

Containers and Managed Services

Devdatta Kulkarni

Problem

- How to Bind application containers with Managed Services?
  - The application code can start interacting with the Managed Service after the binding is successful



Issues in Binding

- How to specify Managed Service's Connection details to the application container?
  - Per Twelve Factor principles, the connection details should be provided through the “Environment”
    - Solved problem
      - Dockerfile provides ENV mechanism
      - Kubernetes provides way to define Pod Env in Pod Yaml / Deployment Yaml

Issues in Binding: Authorization

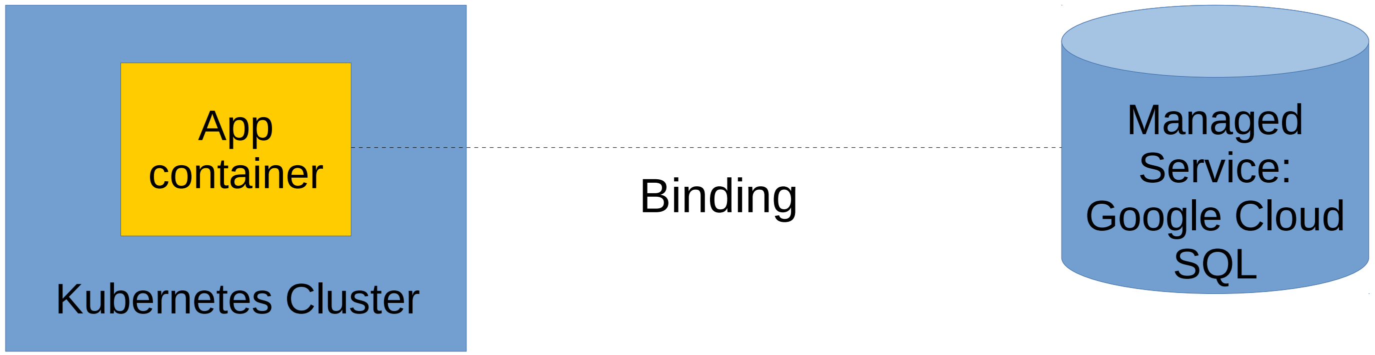
- Container Authorization:
  - This means granting appropriate permissions to your application containers to perform required actions on the managed instance.
  - The required permissions will depend on the type of the data store and the specific actions that your application containers need to take.

Issues in Binding: Traffic Restriction

- Traffic Restriction:
  - This means restricting the network traffic that can reach the managed data store instance.
  - You should only allow traffic that originates from the cluster where your application containers are running to reach the managed instance.
  - Traffic originating from any other source should be disallowed.

Binding Approaches

- Approach 1: Manual approach
  - Manually give appropriate permissions to the App Container
  - Manually modify Firewall/security group of Managed Service instance to receive traffic only from Container cluster

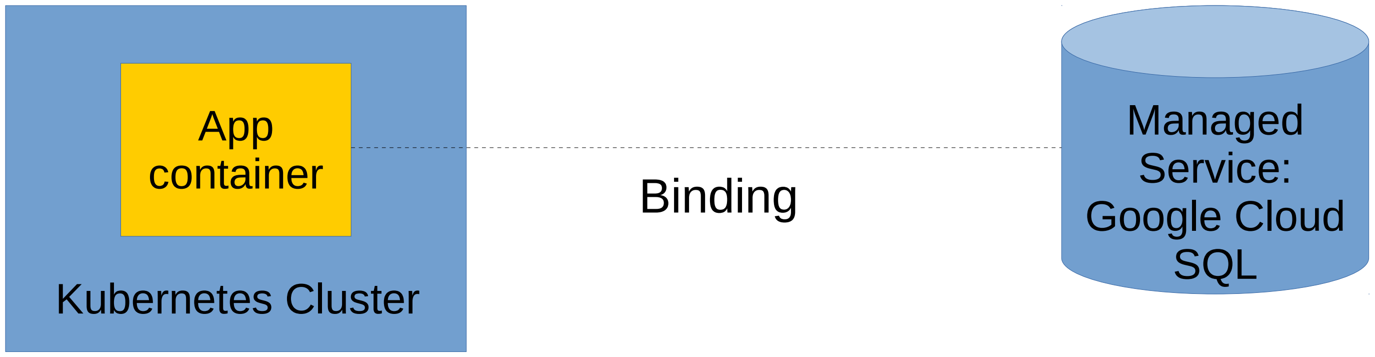


Binding: Manual Approach

- https://github.com/devdattakulkarni/CloudComputing/blob/master/Containers/Kubernetes-examples/GCP/steps.txt
- gcloud container clusters create --machine-type=g1-small --num-nodes=1 testcluster1
- gcloud sql instances create instance1 --tier=db-f1-micro --authorized-networks=0.0.0.0/0
- gcloud sql users set-password root % --instance instance1 --password 'testpass123!@#'
- sudo apt-get install mysql-client
- gcloud sql instances list
- mysql -h <IP-address-of-SQL-Instance> --user=root --password='testpass123!@#'
- mysql>create database testdb;
- gcloud sql instances create instance1 --tier=db-f1-micro --authorized-networks=<CIDR Address of testcluster1>
- Edit greetings-deployment.yaml and set the environment variables
- kubectl create -f greetings-deployment.yaml

Binding Approaches

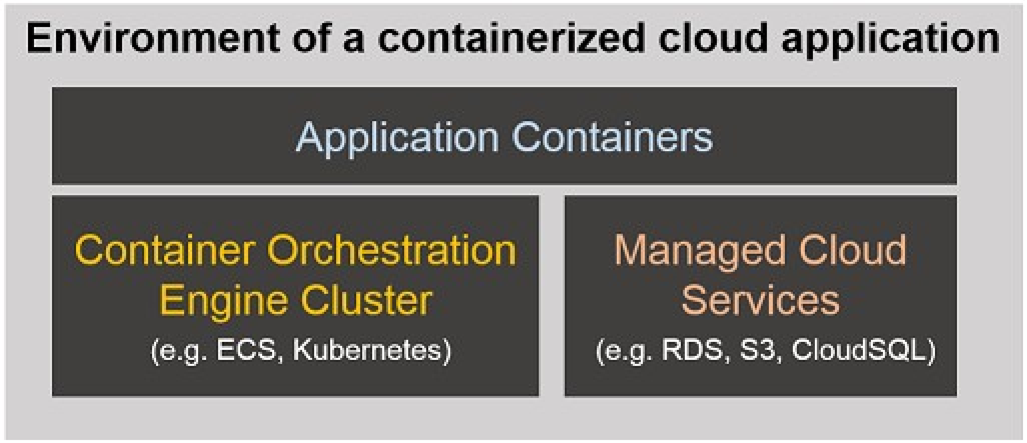
- Approach 2: Automated End-to-end approach
  - Essentially, automate the manual approach
    - Give appropriate permissions to the App Container
    - Modify Firewall/security group of Managed Service instance to receive traffic only from Container cluster



Automated End-to-end Approach

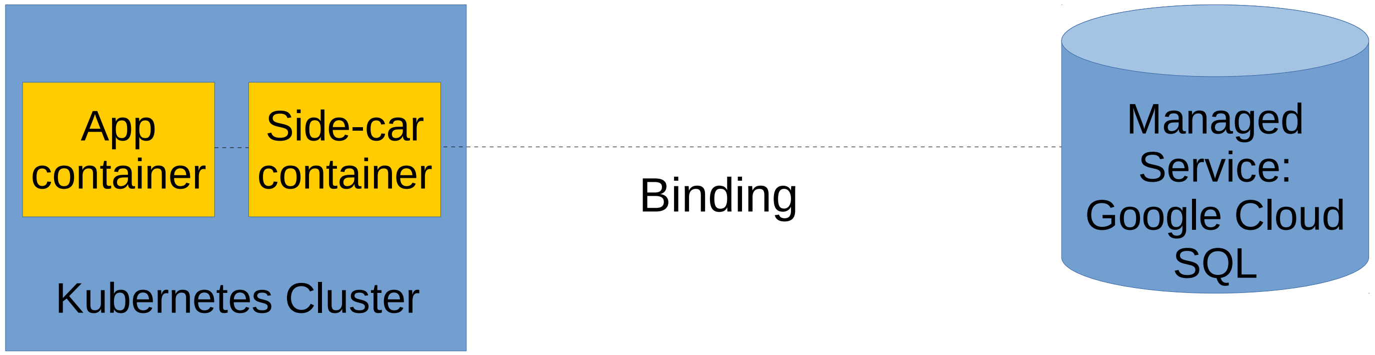
- CaaStle
  - https://github.com/cloud-ark/caastle
  - https://github.com/devdattakulkarni/CloudComputing/blob/master/Containers/Kubernetes-examples/CaaStle/steps.txt

- Abstractions
  - Env
  - App



Binding Approaches

- Approach 3: Sidecar proxy container
  - Run a container that acts as a proxy to the Managed Service
    - Give appropriate permissions to the proxy container
    - App container communicates with the proxy container
    - Modify Firewall/security group of Managed Service instance to receive traffic only from Container cluster



Sidecar Proxy Container

- https://github.com/GoogleCloudPlatform/kubernetes-engine-samples/tree/master/cloudsql

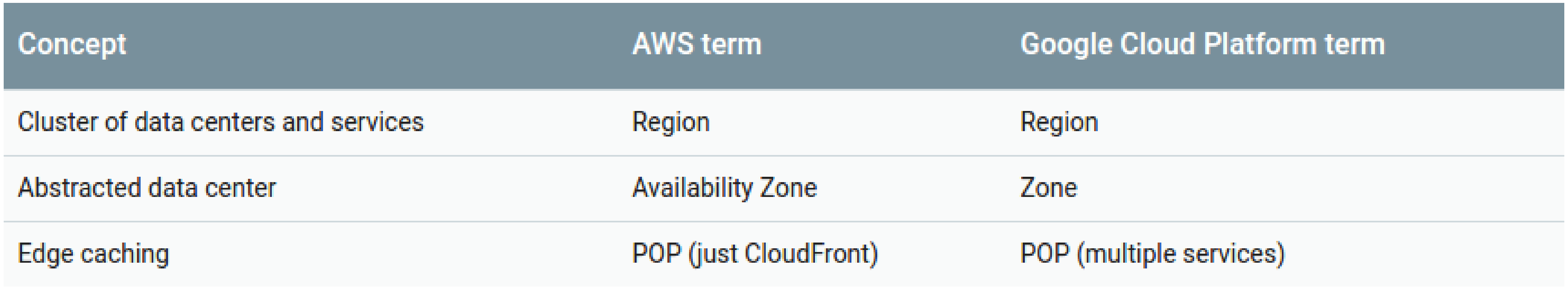
- Design patterns for container-based distributed systems
  - https://www.usenix.org/system/files/conference/hotcloud16/hotcloud16_burns.pdf

Google Cloud Platform (GCP)

Devdatta Kulkarni

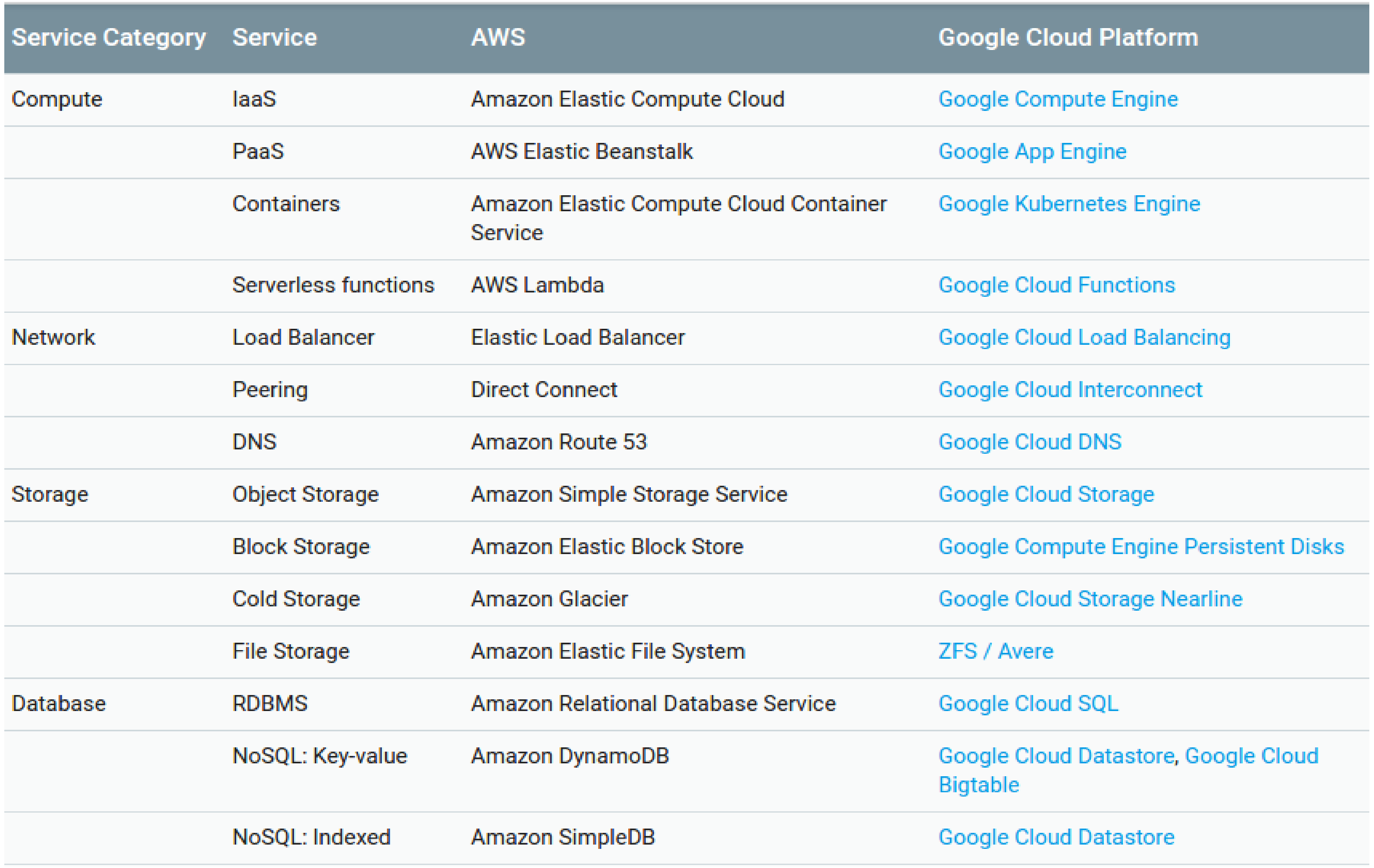
How does GCP compare with AWS?

https://cloud.google.com/docs/compare/aws/



Concept	AWS term	Google Cloud Platform term
Cluster of data centers and services	Region	Region
Abstracted data center	Availability Zone	Zone
Edge caching	POP (just CloudFront)	POP (multiple services)

How does GCP compare with AWS?



Service Category	Service	AWS	Google Cloud Platform
Compute	IaaS	Amazon Elastic Compute Cloud	Google Compute Engine
	PaaS	AWS Elastic Beanstalk	Google App Engine
	Containers	Amazon Elastic Compute Cloud Container Service	Google Kubernetes Engine
	Serverless functions	AWS Lambda	Google Cloud Functions
Network	Load Balancer	Elastic Load Balancer	Google Cloud Load Balancing
	Peering	Direct Connect	Google Cloud Interconnect
	DNS	Amazon Route 53	Google Cloud DNS
Storage	Object Storage	Amazon Simple Storage Service	Google Cloud Storage
	Block Storage	Amazon Elastic Block Store	Google Compute Engine Persistent Disks
	Cold Storage	Amazon Glacier	Google Cloud Storage Nearline
	File Storage	Amazon Elastic File System	ZFS / Avere
Database	RDBMS	Amazon Relational Database Service	Google Cloud SQL
	NoSQL: Key-value	Amazon DynamoDB	Google Cloud Datastore, Google Cloud Bigtable
	NoSQL: Indexed	Amazon SimpleDB	Google Cloud Datastore

GCP IAM

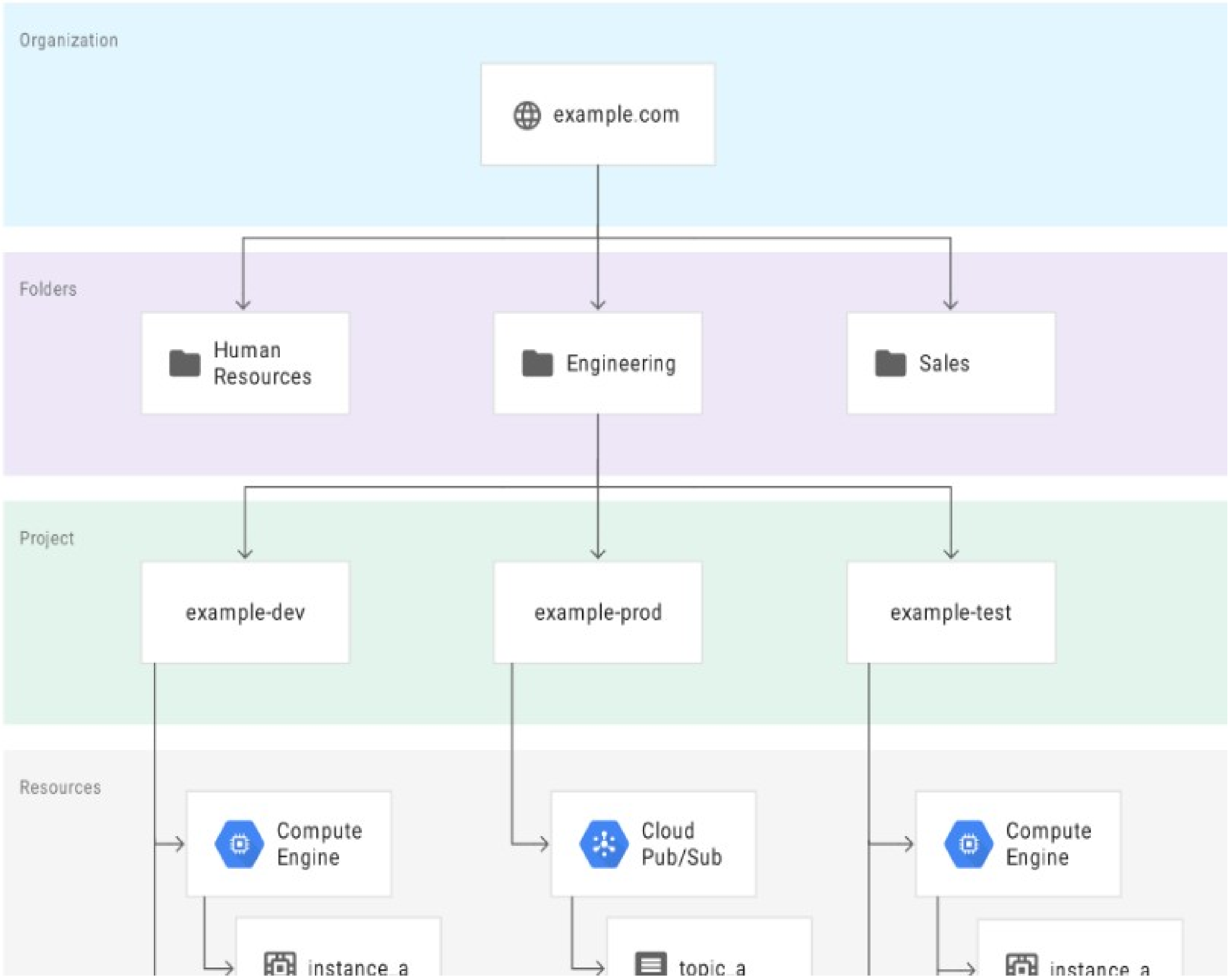
- https://cloud.google.com/iam/docs/overview

- Enables granting access to GCP
  - Who (identity) has what (role) access for which resource
- Identity (who)
  - Google account
    - Person
  - Service account
    - For programmatic access

GCP IAM: Resources

- Resources
  - Project
  - Compute Engine Instances
  - Cloud SQL Instances
  - Etc.
- Hierarchical organization of Resources
  - Project is the top-level Resource
  - All other resources are created under the Project

Resource Hierarchy



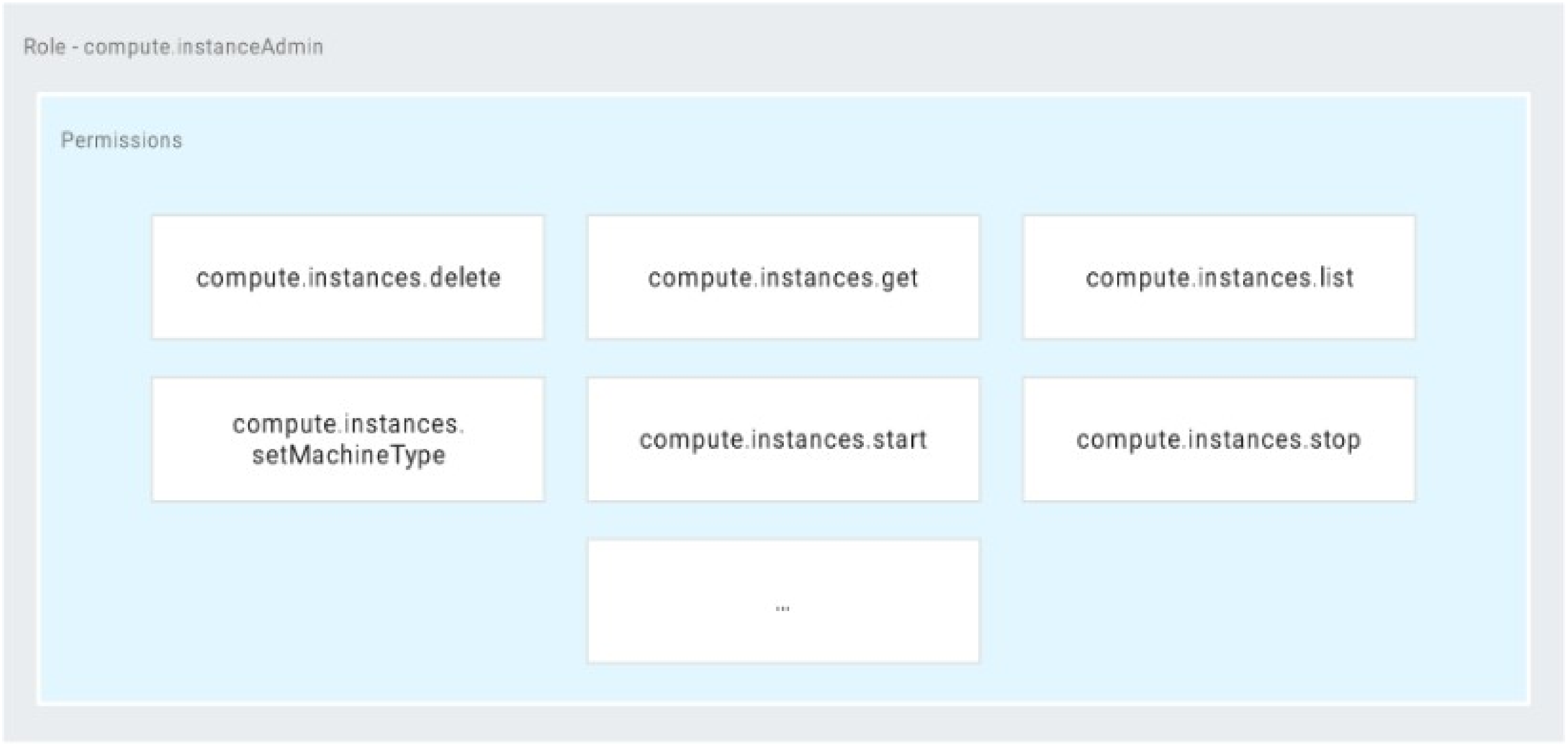
GCP IAM: Permissions

- Permissions
  - Determine what operations are allowed on a resource
  - Represented as:
    - <service>.<resource>.<verb>
  - Permissions usually correspond 1-to-1 with the REST actions on a resource
    - Caller of a REST action needs the corresponding permisisons

GCP IAM: Roles

- Roles
  - A role is a collection of permissions
  - Permissions ***cannot*** be assigned to Users directly
    - This is one of the main differences from AWS IAM model
  - Users gain permission through roles
    - When a role is granted to a User, all the permissions of that role are available to that User

GCP IAM: Roles



GCP IAM: Roles

- Primitive roles
  - Owner, Editor, Viewer
- Predefined roles
  - Roles that give fine-grained permissions that the primitive roles cannot satisfy
- Custom roles
  - Roles that you can create if predefined roles do not satisfy your needs

GCP IAM: Policy

- IAM Policy
  - Grants/binds roles to IAM members
  - Attached to resource

```
{
  "bindings": [
   {
     "role": "roles/storage.objectAdmin",
     "members": [
       "user:alice@example.com",
       "serviceAccount:my-other-app@appspot.gserviceaccount.com",
       "group:admins@example.com",
       "domain:google.com" ]
   },
   {
     "role": "roles/storage.objectViewer",
     "members": ["user:bob@example.com"]
   }
   ]
}
```

GCP IAM: Policy Hierarchy

- Policy can be defined at any level in Resource hierarchy
- Resources inherit the policies of the parent resource
- If a policy is set at Project-level, it is inherited by all its child resources
- Child policies cannot restrict access granted at a higher level
- Effective policy at a resource = Union of policies defined at the resource and inherited policies

Configuring User Access

- Manual setup
  - gcloud auth login
    - This will Open up browser window asking you to grant permission to gcloud cli to connect to your Google Cloud Account
    - Once you grant the permission, a Token is displayed
    - You will have to copy and paste this Token in your command window
  - Note that the Token is valid for one hour
  - If you need to perform deployment again after one hour, you will have to go through 'gcloud auth login' again

Configuring User Access

- Non-time bounded access
  - Option 1: Achieved through 'Service Account'
    - Create a 'Service Account'
    - Grant necessary Roles to it
    - Download the Service Account Key file (JSON format)
    - Configure gcloud CLI to use Service Account Key file
      - Set a specific environment variable
      - https://cloud.google.com/docs/authentication/production#setting_the_environment_variable
  - https://cloud.google.com/docs/authentication/getting-started#auth-cloud-implicit-python

Configuring User Access

- Non-time bounded access
  - Option 2: Automated Renewal
    - Use CaaStle
      - https://cloud-ark.github.io/caastle/docs/html/html/cloud_auth.html

Google Cloud SQL

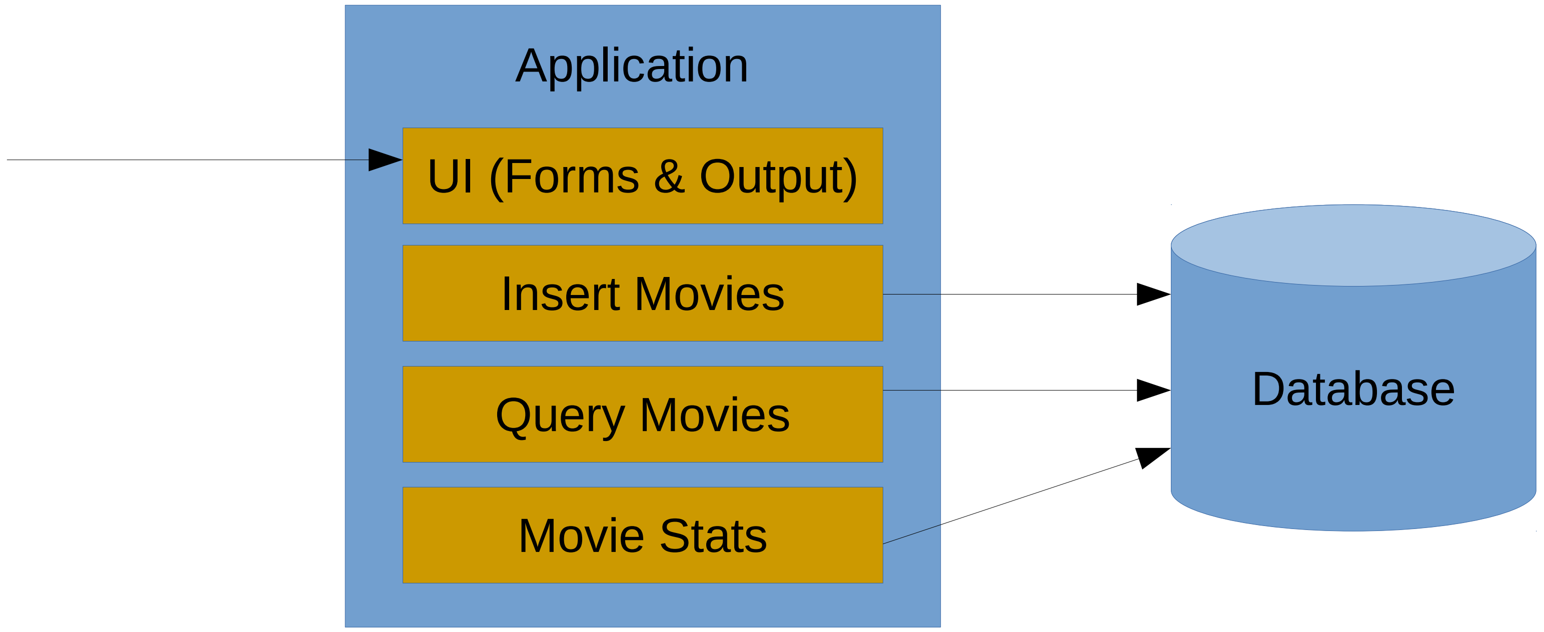
- https://cloud.google.com/sql/docs/mysql/

Microservices

Devdatta Kulkarni

What are Microservices?

- Traditional Application Architecture
  - Monolithic
    - All the functionality embedded in the same code
      - Independent of whether the application is containerized or not



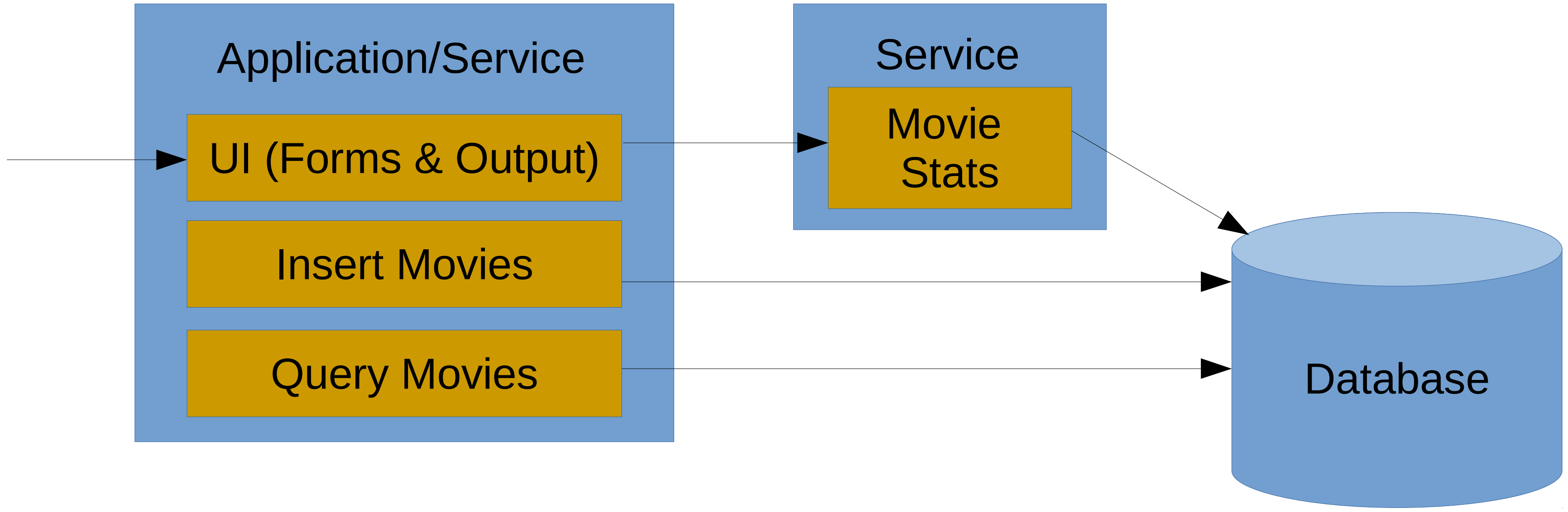
Pros/Cons of Monolithic Architecture

- Easy to develop
- Easy to deploy

- Need to be developed by the same team
- Typically, entire application is developed in a single language
- Independent evolution of application components/features is not possible
- Independent scaling of application components is not possible

Microservice Architecture

- Break up application functionality into separate services that are accessible over HTTP/REST



Pros/Cons of Microservice Architecture

- Separate evolution of application features/ components
  - Different teams can work on different features
  - Different scaling requirements of different services are easy to support
- Services can be written in different languages

- Overhead when beginning application development

Why Microservices now?

- Containerization has made it possible to easily package different application components as independent deployable units

- Systems like Kubernetes make it possible to easily run and scale containers
  - PaaS systems like Elastic Beanstalk and Google App Engine are good for managing and operating monolithic applications but not so much for operating containers

Microservice Architecture Challenges

- How do services discover and bind to one another?
  - In Kubernetes this is achieved through the <service-name>:<target-port> defined in the Service object that is defined for each Microservice
    - https://piotrminkowski.wordpress.com/2017/03/31/microservices-with-kubernetes-and-docker/

- What happens when one service is failing (or is not responsive)?
  - Robust programming of each Service is needed
    - Each service needs to programmed to assume that other services may not be available or may encounter failures

Microservices: Meta points

- How to define Microservice boundaries?
  - No specific rule!
  - Depends on the application features, teams involved, deployment schedules of application (and its features)
- Evolution
  - Typically, you will start with Monolithic application architecture
  - Overtime the application architecture *may* transition into a Microservice architecture
    - Not every application needs to be architected as a Microservice architecture!

PaaS vs CaaS

Devdatta Kulkarni

Platform-as-a-Service (PaaS)

- Elastic Beanstalk
- Google App Engine
- Heroku
- Cloud Foundry
- IBM BlueMix

Container Orchestration as-a-Service (CaaS)

- Kubernetes
- Amazon Elastic Container Service (ECS)
- Docker Swarm
- Apache Mesos

Similarities

- Both type of systems provide following basic Platform functions:
  - Run/Deploy application
    - Rollback to previous version
  - Bind application to managed services like databases
  - Scaling (up/down) of application instances
  - Routing of outside requests to appropriate application instances
  - Load balancing of incoming traffic
    - Deciding what percentage of traffic should be sent to which application instance

Differences

- Deployment
  - PaaSes deploy application starting from application source code
  - CaaSes deploy application containers
    - Building of application containers typically needs to be done separately
- Managed service handling
  - PaaSes typically provision managed service instances (e.g.: RDS instance) and then bind application to it
  - CaaSes mostly focus on application container deployment but provide mechanism for binding
    - Provisioning of managed service instances needs to be done separately
- Local development support
  - PaaSes have been lacking in good local development support
  - Docker and container technology makes local development really simple

Differences

- Application Architecture
  - PaaSes are more suitable if application architecture is monolithic
  - CaaSes are more suitable for both monolithic as well as microservice based application architectures

- End-to-End functionality
  - PaaSes typically provide end-to-end functionality required by applications
    - DNS name, Load balancer, Database
  - CaaSes require you to assemble the required pieces
    - Separately deploy Load balancer, Separately create DNS name and associate it with Load balancer, provision Database, etc.

Which option would you choose? Why?

- You want to start from application code and don't have to worry about writing Dockerfiles

- Your application consists of separate components that are developed by different teams

- Your application needs to be deployed at the end of every week

- Your application needs to be scaled up and rolledback to a previous version

- You are still developing your application and want to deploy and test it during this phase

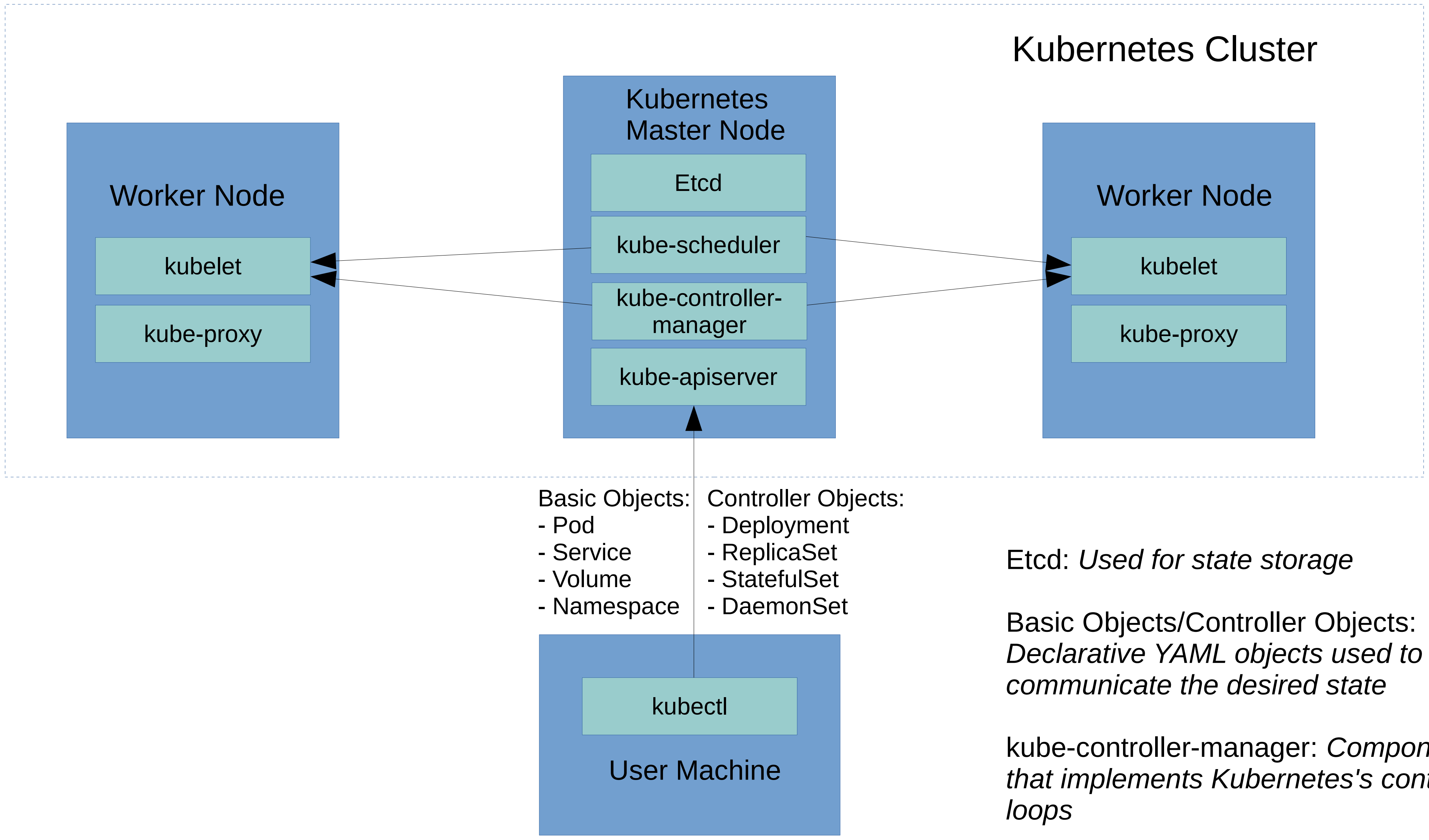
Kubernetes

Devdatta Kulkarni

What is Kubernetes?

- Container Orchestration System developed by Google
  - Open sourced in 2016
  - https://github.com/kubernetes/kubernetes
- Written in Golang
- https://kubernetes.io/docs/tutorials/kubernetes-basics/

Architecture

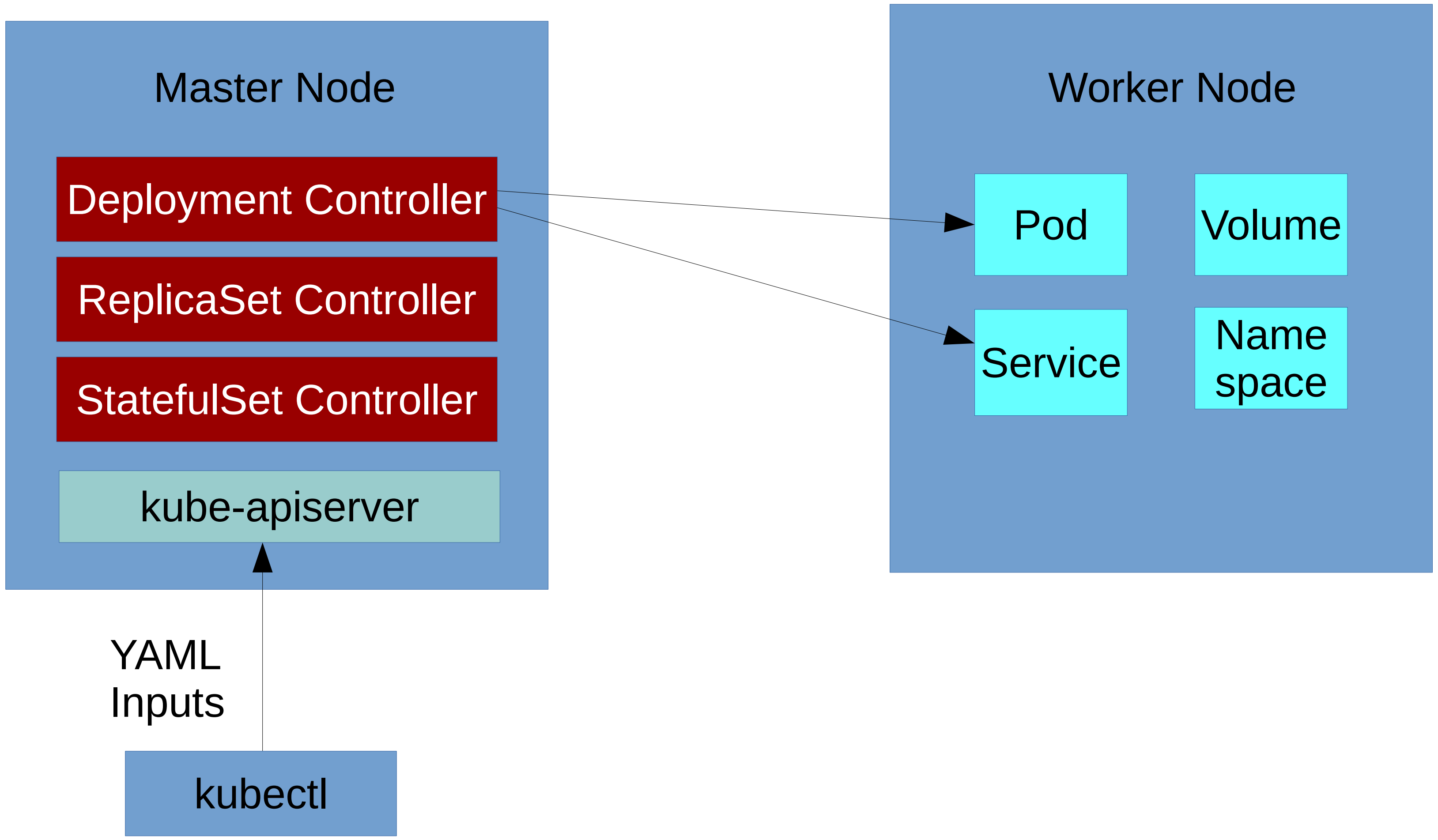


Etcd: *Used for state storage*

Basic Objects/Controller Objects: *Declarative YAML objects used to communicate the desired state*

kube-controller-manager: *Component that implements Kubernetes's control loops*

Architecture



Kubernetes Control Loop

- What is the desired state?
- What is the current state?
- Is desired_state == current_state?
  - If not, make required changes so that the current state becomes same as the desired state.
    - This is called as the *reconciliation action*
- All the Kubernetes Controllers continuously execute this control loop

Basic Objects/Abstractions

- Pod
- Service
- Volume
- Namespace

Pod

- A Pod represents a unit of deployment

- A Pod definition contains following:
  - an application container
  - Storage resources required by the container
  - A Unique network IP address
  - Options that govern how the container should run

- https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/

Example of a Pod Object

```
apiVersion: v1
kind: Pod
metadata:
  name: greetings
  labels:
    app: greetings
spec:
  containers:
    - image: us.gcr.io/cloudark-test-gke/greetpod:1512924392448
      name: greetpod
      ports:
      - containerPort: 5000
      env:
        - name: "PASSWORD"
          value: somepassword
        - name: "DB"
          value: testdb
        - name: "HOST"
          Value: 35.225.61.225
        - name: "USER"
          value: root
```

- Save this as: greetings-pod.yaml
- Create the Pod Object:
  - kubectl create -f greetings-pod.yaml

Pod creation - Behind the scene

- kubectl create -f greetings.yaml
  - Executed on User Machine
  - Corresponding request received by kube-apiserver
    - kube-apiserver stores the Pod Object YAML in Etcd
- kube-apiserver hands over the request to kube-controller-manager
  - kube-controller-manager runs the basic control loop
    - What is the desired state?
      - A new Pod with name 'greetings' needs to be created
    - What is the observed state?
      - A Pod with name 'greetings' does not exist in the cluster
    - Reconciliation action
      - Create a Pod in the cluster

Service

- A Service object defines a logical set of Pods and a policy by which to access them
- The set of Pods targeted by a service is usually determined by a *LabelSelector*
- Although each Pod has a unique IP address, those IPs are not exposed outside the cluster without a Service
- Services allow your application to receive traffic
- https://kubernetes.io/docs/tutorials/kubernetes-basics/expose-intro/

Example of a Service Object

```
kind: Service
apiVersion: v1
metadata:
  name: greeting-service
spec:
  selector:
    app: greetings
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
  type: NodePort
```

- Save this as: greetings-service.yaml
- Create the Service Object:
   - kubectl create -f greetings-service.yaml

Service

- Services can be exposed in different ways by specifying a type in the ServiceSpec:
  - ClusterIP: This exposes the Service on an internal IP in the cluster. The Service is only reachable from within the cluster
  - NodePort: Exposes the Service on the same port of each selected Node in the cluster. From outside the cluster the service is accessible using:
    - <NodeIP>:<NodePort>
  - LoadBalancer: Creates an external load balancer in the underlying cloud and assigns a fixed, external IP of the Service
  - ExternalName: Exposes the Service using an arbitrary name

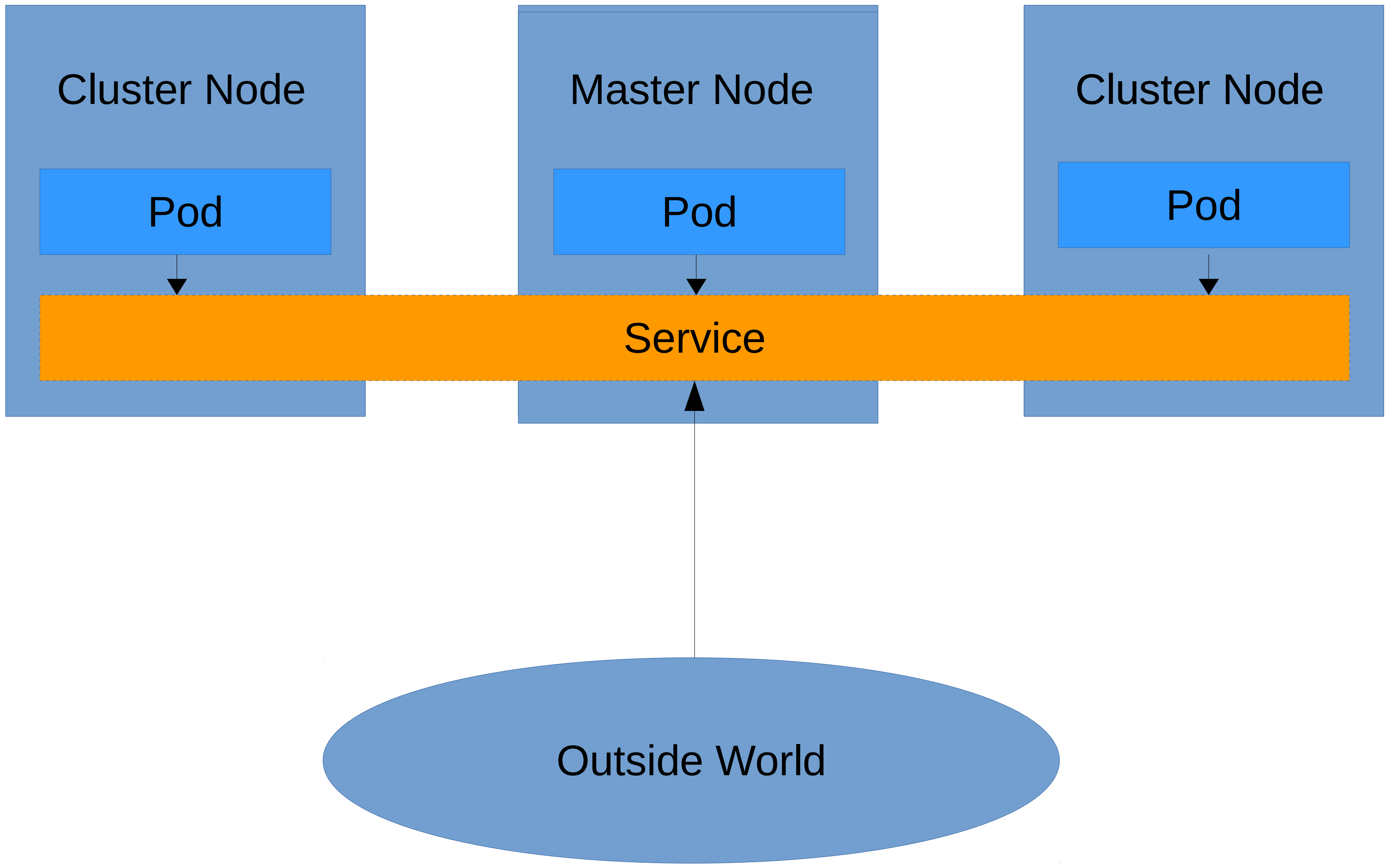
So, how to deploy a container?

Option 1: Create a Pod and Expose it using a Service

- kubectl create -f greetings-pod.yaml
- kubectl create -f greetings-service.yaml

- Make sure that the label inside greetings-pod.yaml matches the selector in greetings-service.yaml

Deploying container: Pod + Service



Problems with Pod+Service option

- Need to create and manage the Pod yaml and the Service yaml
- Need to keep track of label and selector across two different yaml files
- How to create more than one Pod with a given label?
- What if we want to deploy new versions of a particular Pod?
- What if we want to revert to an old version of a particular Pod?

- Enter the Deployment controller
  - A Deployment Object and the corresponding Deployment controller helps address above issues

Deployment Controller

- A Deployment Controller provides a:
  - Declarative model for creating/updating Pods
  - Way to expose the Pods to outside world
  - Supports advanced use-cases

- https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Deployment Controller Use-cases

- Declare and create new Pods
- Scale up the Deployment to facilitate more load
- Rollback to an earlier Deployment
- Pause the Deployment

nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

- A Deployment named nginx-deployment is created, indicated by 'metadata.name' field
- The Deployment creates three replicated Pods, indicated by the 'replicas' field
- The 'selector' field defines how the Deployment controller will find which Pod to manage. It is set to match any Pod that has 'app: nginx' label
- The Pod is specified as a 'template' in the Deployment YAML
- The Pod label ('template.labels') defines the label for the Pods. This needs to match 'selector.matchLabels' field
- The Pod spec defines the container that will be used when the Pod is created

greetings-deployment.yaml

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: greetings-deployment
  labels:
    app: greetings
spec:
  replicas: 1
  selector:
    matchLabels:
      app: greetings
  template:
    metadata:
      labels:
        app: greetings
    spec:
      containers:
      - name: greetings1
        image: gcr.io/cloudark-test-
gke/greetings:v1
        ports:
        - containerPort: 5000
        env:
          - name: "PASSWORD"
            value: testpass123!@#
          - name: "DB"
            value: testdb
          - name: "HOST"
            value: 35.225.61.225
          - name: "USER"
            value: root
```

- A Deployment named greetings-deployment is created, indicated by 'metadata.name' field
- The Deployment creates one of the Pod, indicated by the 'replicas' field
- The 'selector' field defines how the Deployment controller will find which Pod to manage. It is set to match any Pod that has 'app: greetings' label
- The Pod is specified as a 'template' in the Deployment YAML
- The Pod label ('template.labels') defines the label for the Pods. This needs to match 'selector.matchLabels' field
- The Pod spec defines the container that will be used when the Pod is created

So, how to deploy a container?

Option 2: Use a Deployment object

- kubectl create -f greetings-deployment.yaml
- kubectl expose deployments --port=80 --target-port=5000 --type=LoadBalancer greetings-deployment

- The Deployment controller will internally create a 'Service' object for exposing the Pods defined in Deployment YAML

CI/CD using Kubernetes

- Problem: Regular Kubernetes deployments change the IP address of the application
  - Each Service Object gets a different IP address for each deployment
  - In CI/CD setup, keeping IP address constant of an application is crucial!
  - How to achieve this?
- Answer: Use Kubernetes Ingress Object (Kind)

Kubernetes Ingress on GKE

Create a static IP address for our application

```
- gcloud compute addresses create greetings-ip --global
```

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: greetings-ingress
  annotations:
    kubernetes.io/ingress.global-static-ip-name: greetings-ip
spec:
  backend:
    serviceName: greetings-deployment
    servicePort: 5000
```

Namespaces

- https://kubernetes.io/docs/tasks/administer-cluster/namespaces/
- Mechanism that allows using same Kubernetes Cluster for running Pods, Services, Deployments, etc. for different applications
- Users interacting with one namespace do not see the content in another namespace

Namespaces

- https://kubernetes.io/docs/tasks/administer-cluster/namespaces-walkthrough/
- kubectl config view
- kubectl config current-context
- kubectl config set-context dev --namespace=development \
  --cluster=lithe-cocoa-92103_kubernetes --user=lithe-cocoa-92103_kubernetes
- kubectl config use-context dev
- kubectl run snowflake --image=kubernetes/serve_hostname --replicas=2
- kubectl get deployment

```
NAME        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
snowflake   2         2         2            2           2m
```

- kubectl get pods -l run=snowflake

```
NAME                         READY     STATUS    RESTARTS   AGE
snowflake-3968820950-9dgr8   1/1       Running   0          2m
snowflake-3968820950-vgc4n   1/1       Running   0          2m
```

Labels

- Mechanism for selecting and accessing Kubernetes Objects

- Used by controllers in performing reconciliation actions as part of the Control loop
  - Remove a Pod from receiving traffic
    - Remove “frontend” label from the Pod
  - Rollback a deployment
    - Update Deployment Object in which Pod label is set to previous version
      - The Deployment controller will delete the Pods with current label and spin up Pods with the new (previous) label

Volumes

- https://kubernetes.io/docs/concepts/storage/volumes/
- Mechanism that allows Pods to store data in a “filesystem”
  - A Volume is a directory accessible to containers in a Pod
- A Volume is terminated/deleted when the corresponding Pod is terminated

Several Volume Types Available

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- configMap
- csi
- downwardAPI
- emptyDir
- fc (fibre channel)
- flocker
- gcePersistentDisk
- gitRepo
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume

ConfigMap Volume Type

- Special type of Volume typically used for providing configuration data to containers in a Pod
- ConfigMap
  - https://kubernetes-v1-4.github.io/docs/user-guide/configmap/

Cloud Computing: Revisited

Devdatta Kulkarni

Why are Enterprises/Organizations adopting Cloud Computing?

- No upfront investment in hardware resources
  - CapEx vs OpEx
    - Capital Expenditure: Invest capital/money upfront
    - Operational Expenditure: Invest capital/money for operational expenses
    - Same pros/cons as renting vs. owning an asset (house/car)
- More secure (ironic, but true!)
  - Cloud providers have deep security expertise
- Easy to adopt DevOps/Agile software development practices
  - Why?
- Quickly adopt newer technologies
  - Containers, Kubernetes, TensorFlow, Alexa, GPUs, etc.
- Available across the Globe
  - Multi-region support

Why are Enterprises/Organizations adopting Cloud Computing?

- SaaS
  - Providing Software-as-a-Service (SaaS) model to end users is better than requiring installations on their machines
    - Think Google Docs vs. Microsoft Word installed on your machine
      - No hassle of installation, upgrades, virus scanning, etc.
    - Think Gmail vs. installing Mailer Daemon and Mail clients in an enterprise/organizations

Why are Enterprises/Organizations adopting Cloud Computing?

- Other Reasons?

Who is going to Cloud - Some examples

- Wall Street (Public cloud)
  - https://www.brinknews.com/why-wall-street-is-moving-to-the-cloud/
- Capital One (Public cloud)
  - https://medium.com/aws-enterprise-collection/capital-ones-cloud-journey-through-the-stages-of-adoption-bb0895d7772c
- TCS / SUSE (Private cloud)
  - https://www.suse.com/media/presentation/CAS91630_convert_your_datacenter_to_an_enterprise_public_cloud.pdf
- There are many more companies in many different sectors

Cloud Computing Concerns

- Vendor lock-in
  - Once a cloud is chosen by an Organization, it is hard to change (i.e. to go to another cloud)
- Migration
  - How to move existing (in-house) code/data/applications to Cloud?
- Cloud Expertise
  - How to best utilize the Cloud?
    - Each Cloud computing platform is different

Mitigation tactics

- How to address Vendor lock-in?
  - Use multiple clouds
  - Use open source Cloud software such as OpenStack
- How to handle migration challenges?
  - Invest in tools for cloud migration
  - *Do not migrate* everything to cloud!
    - Use a 'Hybrid' strategy – only some applications and workloads are migrated to the Cloud

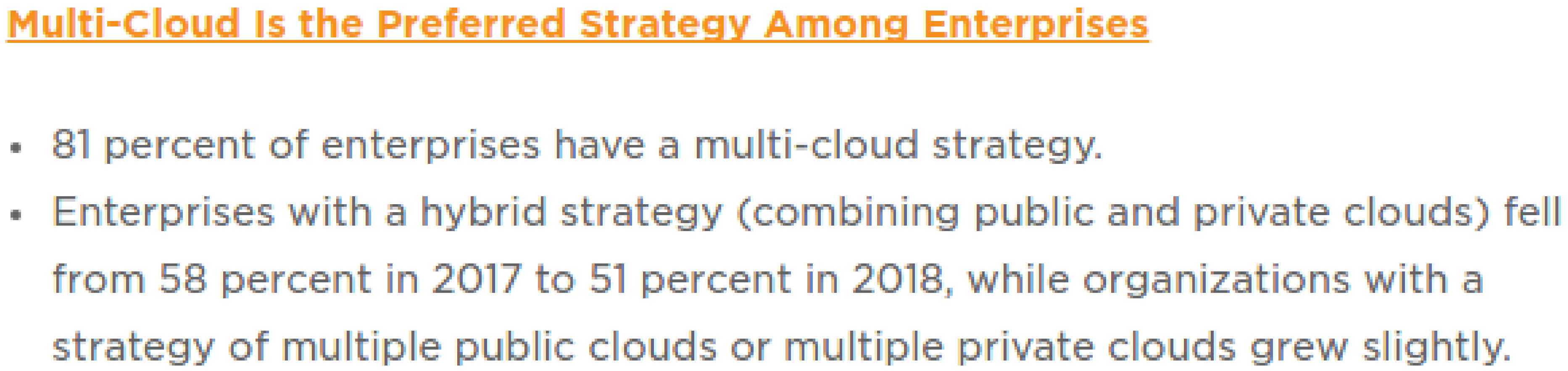
Why Organizations build their own Clouds/Data Centers?

- They want Control
  - http://www.datacenterdynamics.com/content-tracks/colo-cloud/report-target-to-leave-aws-after-amazon-buys-whole-foods/98881.fullarticle
- They are massive
  - https://techcrunch.com/2017/09/15/why-dropbox-decided-to-drop-aws-and-build-its-own-infrastructure-and-network/
  - At massive scale, the cost of using Cloud is expensive
- They have unique requirements
  - https://www.tintri.com/sites/default/files/field/pdf/document/Tintri-Northwestern-University-Case-Study.pdf
  - Unique storage needs

Cloud computing trends - 2018

- Rightscale survey
  - https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2018-state-cloud-survey

Multi-cloud is being preferred



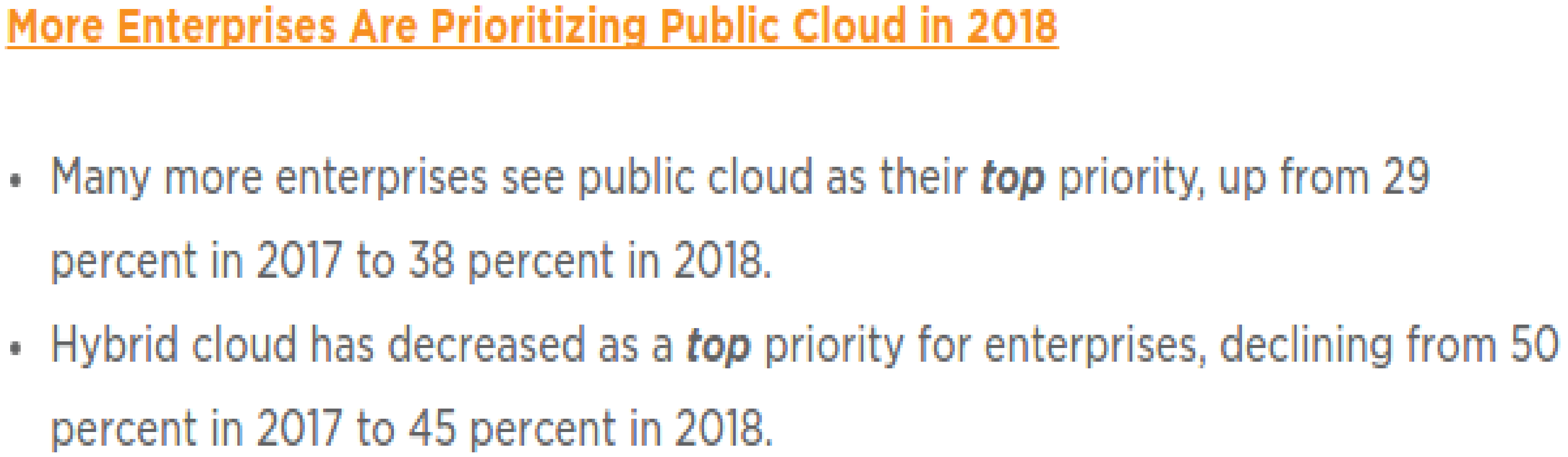
Multi-Cloud Is the Preferred Strategy Among Enterprises

- 81 percent of enterprises have a multi-cloud strategy.
- Enterprises with a hybrid strategy (combining public and private clouds) fell from 58 percent in 2017 to 51 percent in 2018, while organizations with a strategy of multiple public clouds or multiple private clouds grew slightly.

How to do Multi-cloud?

- Use softwares/tools that support Multi-cloud
  - FirstMile
    - Web application deployment on Elastic Beanstalk and Google App Engine
  - CaaStle
    - Containerized application deployment on GKE and Amazon ECS
  - CloudFoundry
    - PaaS that supports several clouds
  - Terraform
    - Infrastructure-as-Code for different cloud providers
  - Gravitant/IBM
    - Cloud Broker software

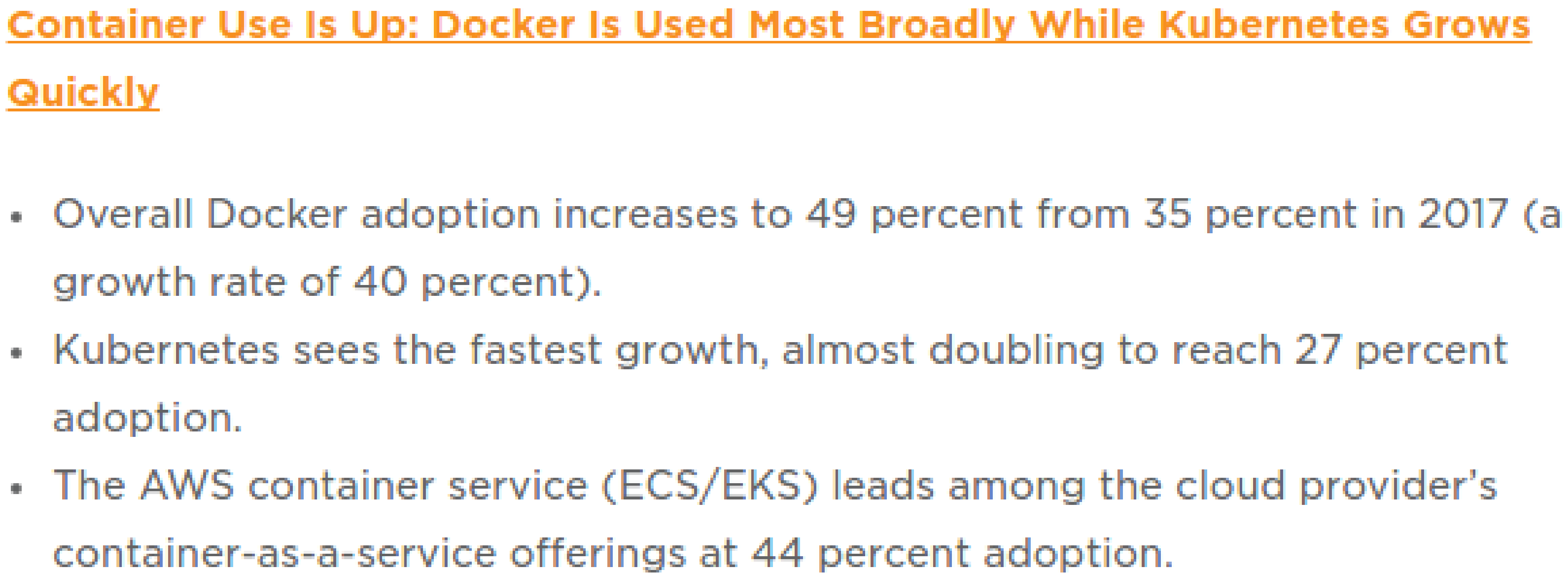
Public clouds top priority



More Enterprises Are Prioritizing Public Cloud in 2018

- Many more enterprises see public cloud as their ***top*** priority, up from 29 percent in 2017 to 38 percent in 2018.
- Hybrid cloud has decreased as a ***top*** priority for enterprises, declining from 50 percent in 2017 to 45 percent in 2018.

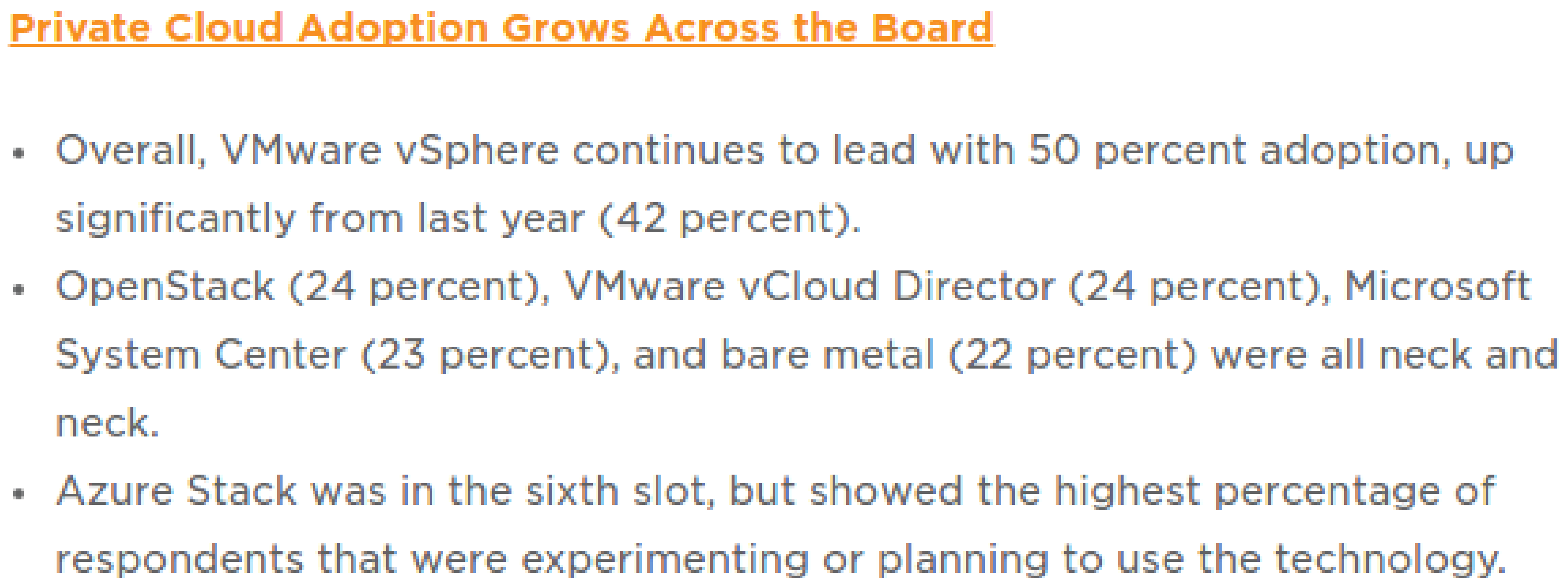
Docker and Kubernetes



Container Use Is Up: Docker Is Used Most Broadly While Kubernetes Grows Quickly

- Overall Docker adoption increases to 49 percent from 35 percent in 2017 (a growth rate of 40 percent).
- Kubernetes sees the fastest growth, almost doubling to reach 27 percent adoption.
- The AWS container service (ECS/EKS) leads among the cloud provider's container-as-a-service offerings at 44 percent adoption.

Private Cloud adoption grows



Private Cloud Adoption Grows Across the Board

- Overall, VMware vSphere continues to lead with 50 percent adoption, up significantly from last year (42 percent).
- OpenStack (24 percent), VMware vCloud Director (24 percent), Microsoft System Center (23 percent), and bare metal (22 percent) were all neck and neck.
- Azure Stack was in the sixth slot, but showed the highest percentage of respondents that were experimenting or planning to use the technology.

Private Cloud

- What is Private Cloud?
  - AWS / GCP kind of setup in an Enterprise's/Organization's own Data center

- How?
  - VMWare vSphere (proprietary)
  - Microsoft System Center (proprietary)
  - OpenStack (Open source)

OpenStack

What is OpenStack?

- Open source software for creating private and public clouds (http://www.openstack.org/)
  - For instance, using OpenStack it is possible for any organization to provide cloud computing capabilities to its users
    - CS department can install OpenStack and all those who have CS accounts can spin up virtual machines as and when needed -- similar to what you get currently from commercial vendors (Amazon, Azure, Rackspace, etc.)
- Founded in 2010 by Rackspace and NASA
  - Many more companies are involved now
    - http://www.openstack.org/foundation/companies/

OpenStack Projects

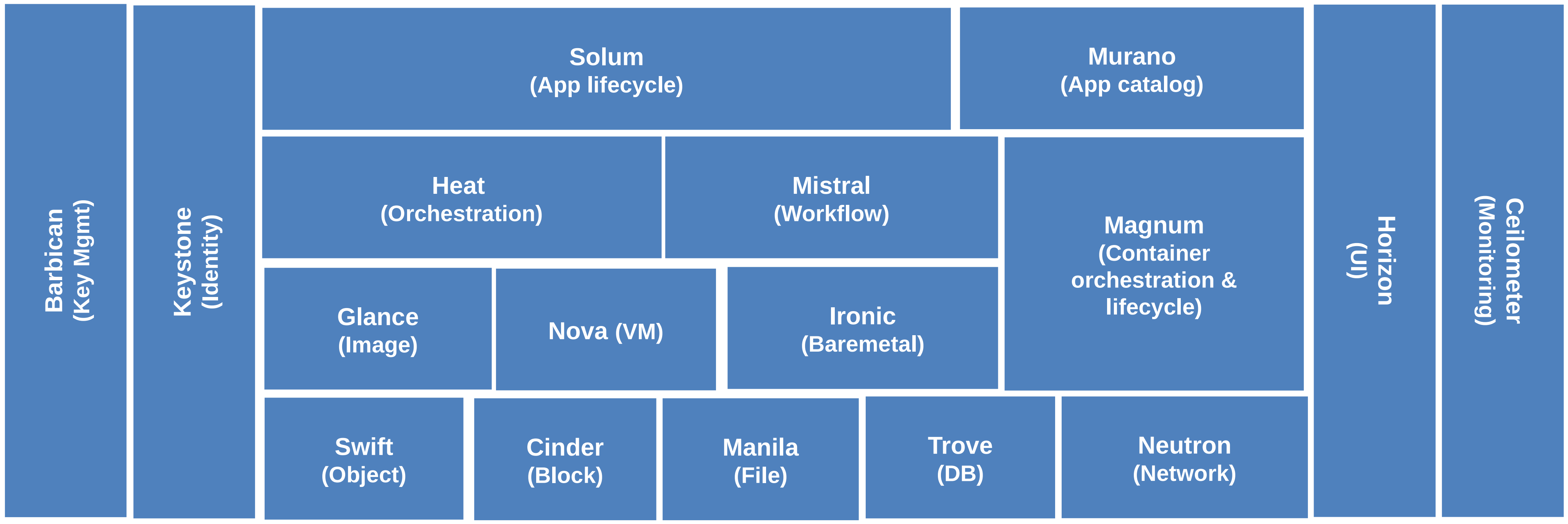
- Compute (Nova)
- Object storage (Swift)
- Image Service (Glance)
- Database Service (Trove)
- Orchestration (Heat)
- Identity Service (Keystone)
- Networking (Neutron)
- Dashboard (Horizon)
- Block storage (Cinder)
- PaaS (Solum)
- More

OpenStack: High-level details

- Code
  - written in Python
  - available on github

- Projects are available in the “openstack” organization on Github
  - https://github.com/openstack

OpenStack: Projects



OpenStack: Development steps

- Sign the CLA
- Setup code submission and code review hooks on your development machine
  - Gerrit is used for code submission and code reviews
- Clone the project that you want to work on
- Work on your patch
- Test the patch using tox
- Submit the patch for review

Contributing to OpenStack

http://www.slideshare.net/devkulkarni/contributing-to-openstack-59590072

CI/CD

Devdatta Kulkarni

CI/CD

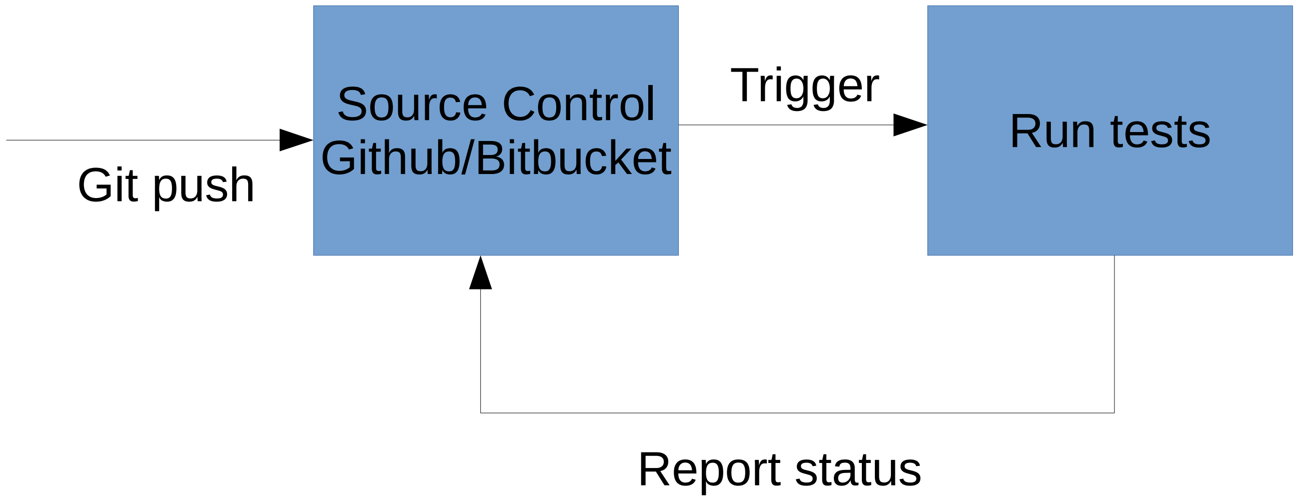
- Continuous Integration / Continuous Delivery
- Continuous Integration / Continuous Deployment

What is Continuous Integration?

- It is the act of continuously testing software

- What kind of tests?
  - Unit tests
  - Integration tests
  - Functional tests
  - Load tests
  - Penetration tests

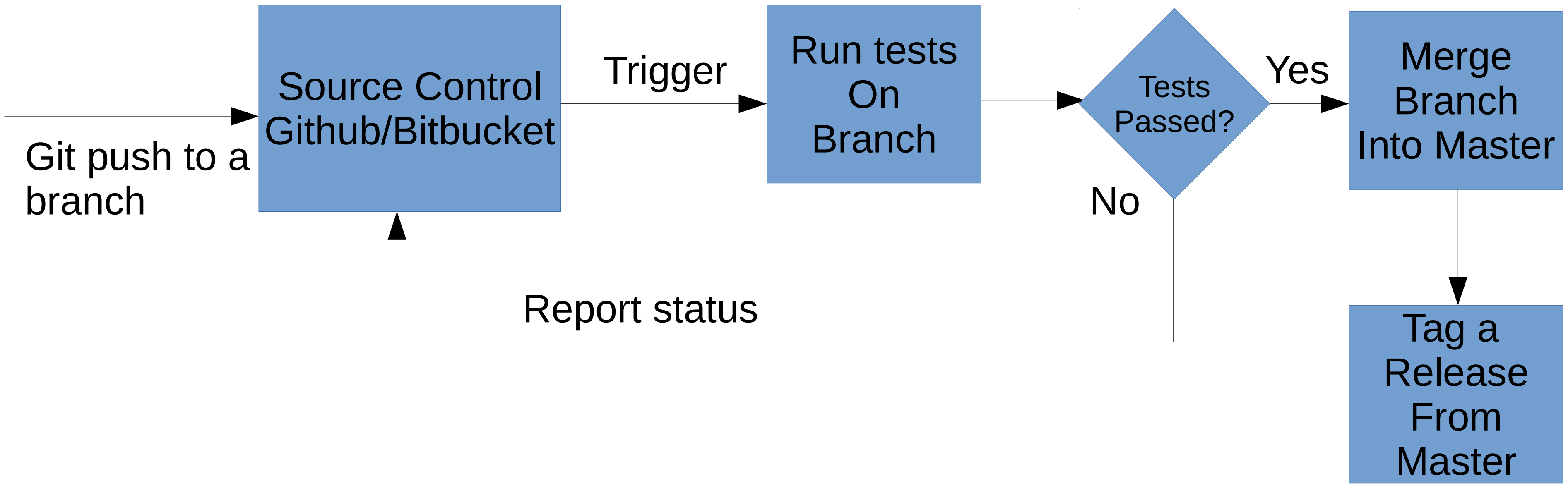
Continuous Integration



What is Continuous Delivery?

- It is the act of tagging a deployable release artifact from well tested code
  - In industry, tagging is also called as “cutting a release”
- What are examples of tagged release artifacts?
  - tar.gz files, war files, docker images, AMIs, etc.
- Where are release artifacts stored?
  - S3, Docker Hub, GCR, ECR, Sonatype's Nexus repository, JFrog's Artifactory, etc.

Continuous Delivery



Pre-merge Gating:
- Work on a Branch
- Run tests on the Branch
- Merge the Branch into Master branch only if tests pass

Advantage of Pre-merge Gating:
- Code is never in a broken state on the Master branch

Pre-merge Gating

- Why “Gating”?
  - The tests act as a “gate” that allows new code into the Master branch

Triggers

- CI/CD actions are triggered through Web hooks
- What is a Web Hook?
  - It is a URL at which CI/CD server receives requests
  - CI/CD server performs actions upon receiving a request
    - Running tests, Tagging a Release, Reporting status, etc.

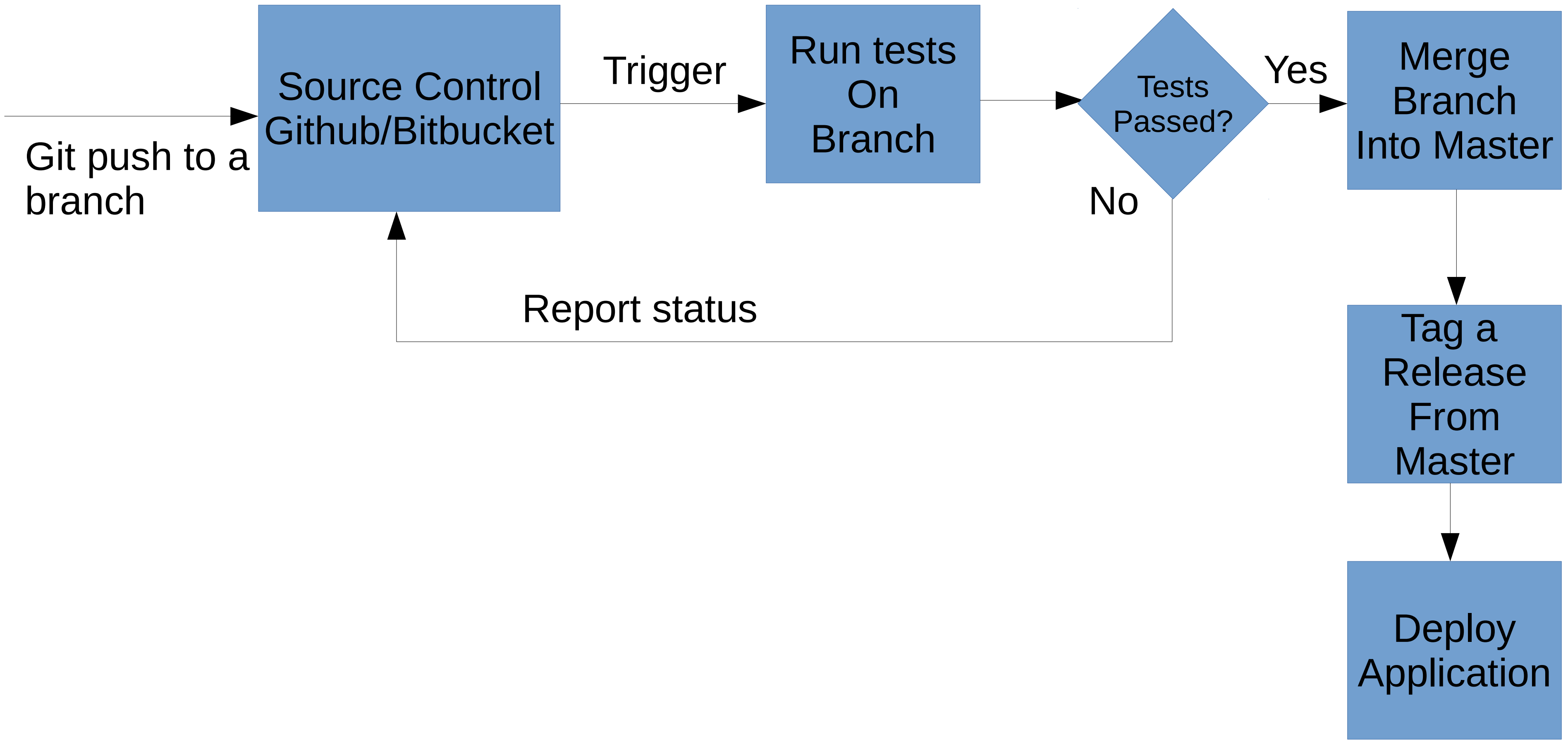
CI/CD Tools

- Jenkins
- CircleCi
- Travis
- TeamCity
- Shippable

What is Continuous Deployment?

- It is the act of deploying the tagged artifact which was tested through continuous integration

Continuous Deployment



Continuously Deployment

- Where is continuous deployment used?
  - Deploying applications to test / staging environments
- Continuous Deployment to Production is rare
  - Usually Production deployments involve a Human in the loop
    - Production releases should not affect customers
    - Usually planned with specific timelines with appropriate communication to customers

Deployment Strategies

- Replacement
- Canary
- Blue-green

Replacement

- Delete currently deployed version of the application
  - If first time deployment then nothing to do
- Deploy new version of the application

- Drawback:
  - Application incurs downtime
    - Application users will see that application is not available during the time it takes to complete the deployment

Canary Deployment

- Deploy new version of the application without removing the old version
  - New version is deployed at a new DNS name (URL)
- Redirect some small percentage of traffic to newly deployed application
- If there are no issues encountered by this 'canary traffic' then
  - Redirect all the traffic to the new version
  - When all the ongoing requests to old version (DNS name/URL) are completed, shutdown the old version

Why Canary?

- Canaries (birds) used by Miners in coal mine to detect carbon monoxide
  - Canaries would die first
- Canary deployments
  - If there is problem with the new version of the application, only the canary traffic will get affected

Blue-green Deployment

- Label the currently deployed version as “Blue”
- Configure load balancer to send traffic to “Blue”

- Deploy new version and label it as “Green”
- Do all the live testing on “Green”
- Once the live testing passes, switch labels
  - This will cause load balancer to start sending traffic to the “Green” version (For the load balancer, this version is the “Blue” version after the switch)
- Keep the new Green (i.e. the old version) around for some time
  - This is required to perform rollback of the new deployment
  - Rollback amounts to switching the labels back

CI + Continuous Deployment Tools

- CI Tool + PaaS
  - Jenkins + Elastic Beanstalk
  - Jenkins + FirstMile
- CI Tool + CaaS
  - Jenkins + Kubernetes
  - Jenkins + CaaStle

Web hooks Revisited

- One of the actions that CI/CD server can perform as part of handling Web hook requests is deploying the application

- What is a Web Hook?
  - It is a URL at which CI/CD server receives requests
  - CI/CD server performs actions upon receiving a request
    - Running tests, Tagging a Release, Reporting status, Deploy application

How are Managed Services Handled?

- How are managed services like Databases handled as part of CI/CD setup?
  - Managed Services are usually kept separate from core CI/CD setup
  - Provisioning and setup of managed services like databases is performed through a process called “Infrastructure creation and configuration management”

Infrastructure creation and Configuration Management

- The act of creating infrastructure elements and configuring them according to application needs

Infrastructure creation and Configuration management: Assignment 5

Infra creation

- Creation of GKE cluster
- Creation of Google Cloud SQL instance

Config Management

- Configuring the 'root' user on Cloud SQL with specific password
- Creation of 'testdb' database on the Cloud SQL instance
- (Additionally) we could have locked down Cloud SQL instance to receive traffic only from the GKE cluster

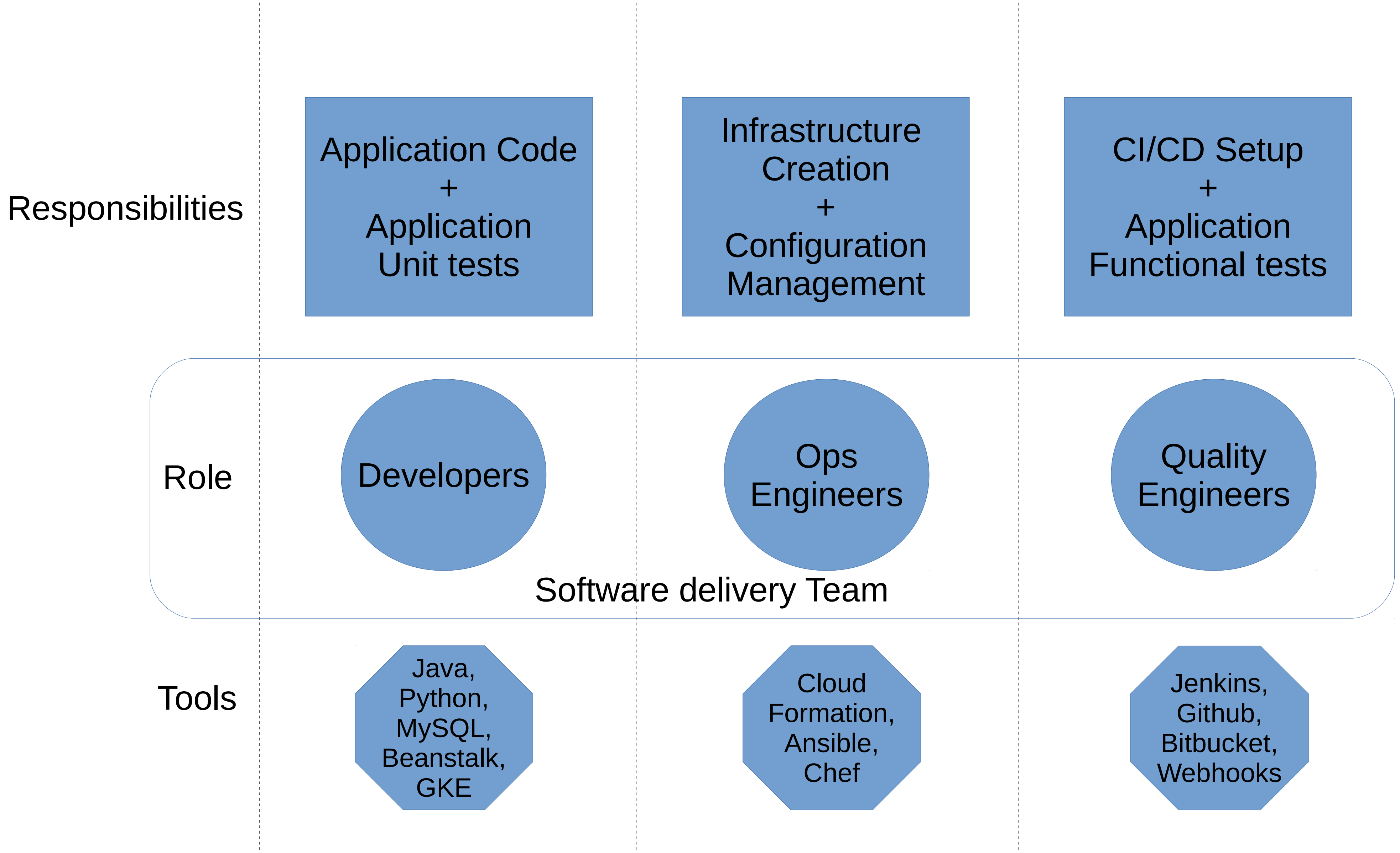
Infra creation & Config Management

- Manual
  - https://github.com/devdattakulkarni/CloudComputing/blob/master/CICD/cicd-steps.txt
- Cloud APIs + Bash scripts
  - Like assignment 1
- CloudFormation + Bash scripts
- Terraform + Bash scripts
- Ansible
- Chef
- Puppet

DevOps Revisited

- DevOps Practice
  - Steps and processes that make use of Configuration Management + CI/CD to continuously release new application features

DevOps: Roles & Responsibilities


Responsibilities
Application Code + Application Unit tests
Infrastructure Creation + Configuration Management
CI/CD Setup + Application Functional tests
Role
Developers
Ops Engineers
Quality Engineers
Software delivery Team
Tools
Java, Python, MySQL, Beanstalk, GKE
Cloud Formation, Ansible, Chef
Jenkins, Github, Bitbucket, Webhooks

DevOps: Why now?

- Why has DevOps practice become popular now?
  - Cloud computing has enabled easy provisioning of resources (Kubernetes clusters, VM instances, Databases, Load balancers, etc.)
  - Docker and container technology has enabled easy packaging of applications into deployable units
  - CI/CD tools have enabled faster code-test-deploy cycle
- Result:
  - Organizations now continuously release new application features using “Agile” method
    - Sharp contrast from old “Waterfall” model of software development, which consisted of following steps:
      - First plan all the features (Usually several months effort)
      - Then develop the features (Several months effort)
      - Deploy the application (several days)

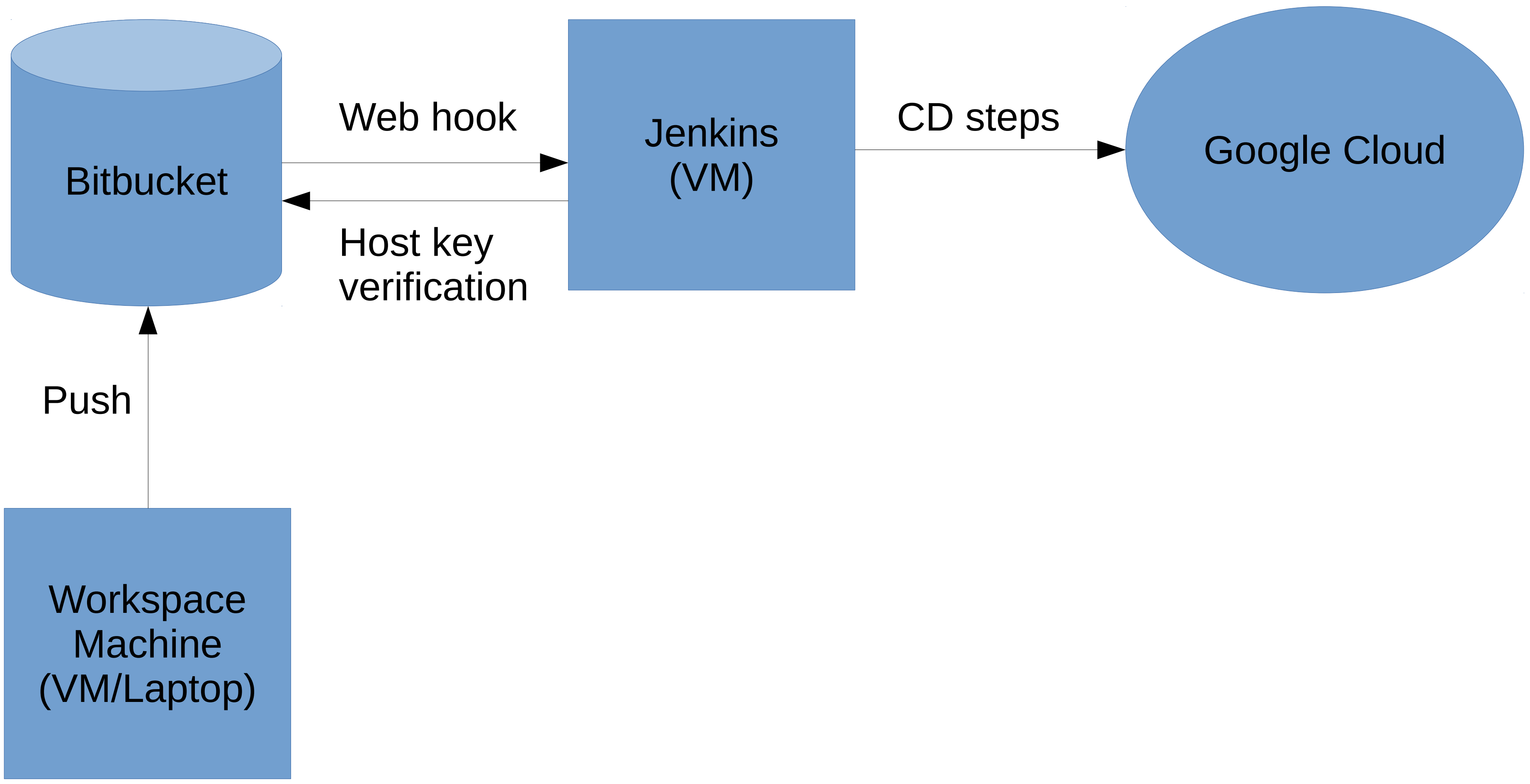
Agile method of software delivery

- Iterative development
  - Two/Three week development plan (typically called “Sprint”)
  - Developers work on application code and writing Unit tests
  - Operations Engineers set up infrastructure along with configuring it as required by application
  - Quality Engineers setup CI/CD and write functional tests
- No Silos
  - All the involved functional roles work very closely together
  - Developers are involved in live support of their applications
    - Being “on call” or having a “pager” duty
      - Typically rotated between the developers every Sprint

CI/CD in Industry

- Facebook
  - 3 deploys per day
  - https://code.facebook.com/posts/270314900139291/rapid-release-at-massive-scale/
- Netflix
  - 16 minutes from code check-in to deployment
  - https://medium.com/netflix-techblog/how-we-build-code-at-netflix-c5d9bd727f15
- Etsy
  - 25 deploys per day in 2012
  - https://www.slideshare.net/beamrider9/continuous-deployment-at-etsy-a-tale-of-two-approaches

Assignment 6

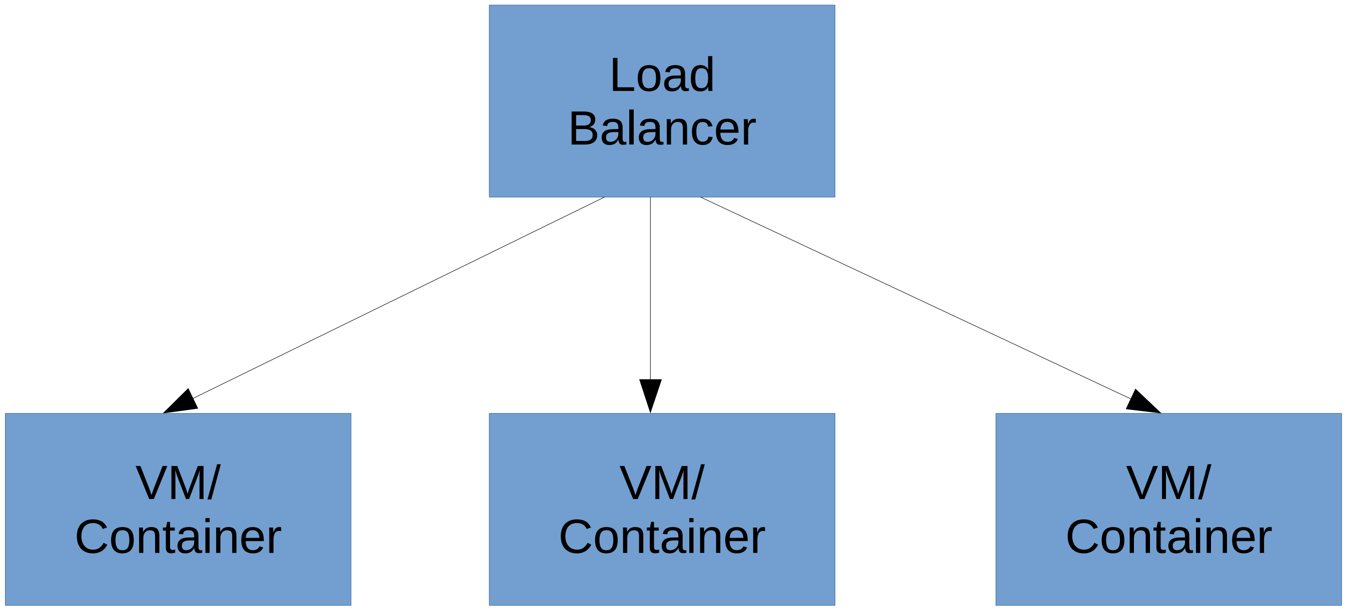


Load Balancing

Devdatta Kulkarni

What is Load Balancing?

- Balance incoming requests across number of machines/VMs/application instances



Basic Operation

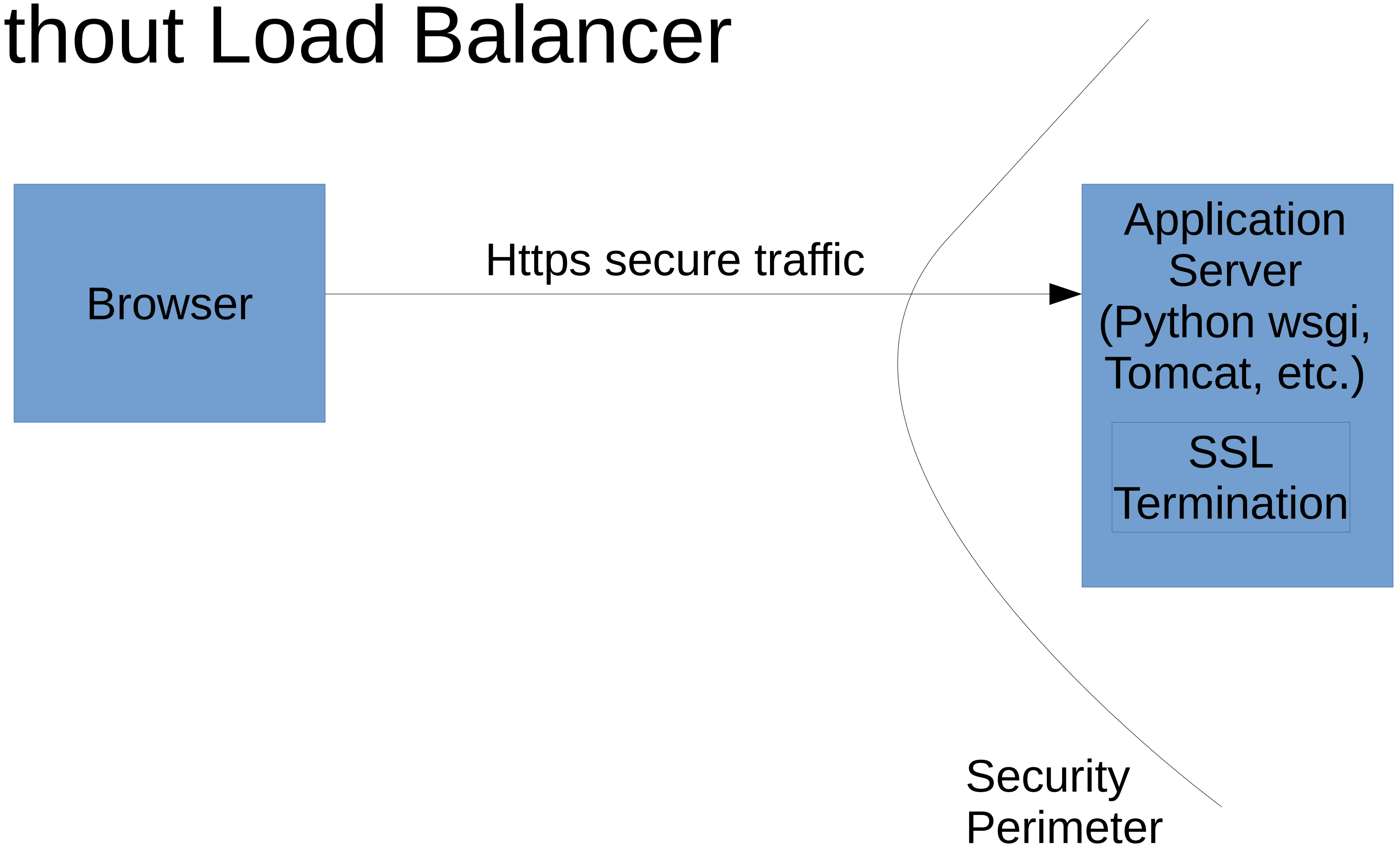
- A redirection Table
  - Input: Paths from where traffic will be received
  - Redirection: List of servers to which the traffic should be redirected
- How to redirect traffic to different backend servers?
  - Several options possible. Following are Nginx options:
    - Round robin
    - Least connection
    - Weighted
    - IP Hash

Advantages of using Load Balancer

- Supporting more clients is possible by sending/redirecting client requests to different application instances
  - Application throughput increases
- Faster responses to client requests
  - Because requests are distributed across application instances, each individual request is handled faster
- Increased availability of the application
  - If one application instance is down, client requests can be sent to other instances
- Scaling of application instances is possible
  - You can add as many application instances behind the load balancer as you want
  - This is called “Horizontal Scaling”
- Application URL does not change
  - It is the IP address / the DNS name of the Load Balancer
- Security in one place
  - SSL Termination can be implemented in the Load Balancer rather than having the application server do it

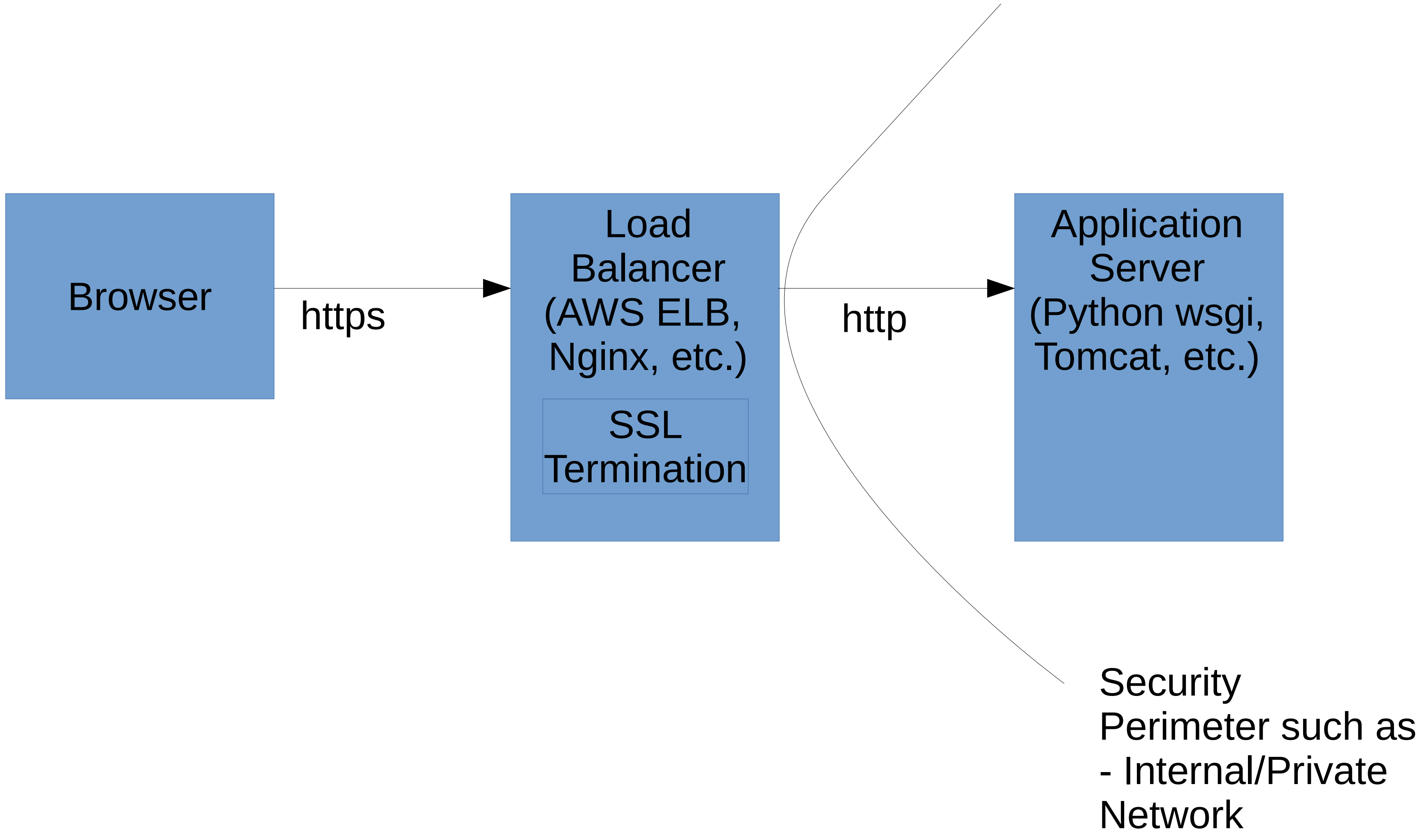
What is SSL Termination?

- It is the process of decrypting https traffic
- Without Load Balancer



What is SSL Termination?

- With Load Balancer

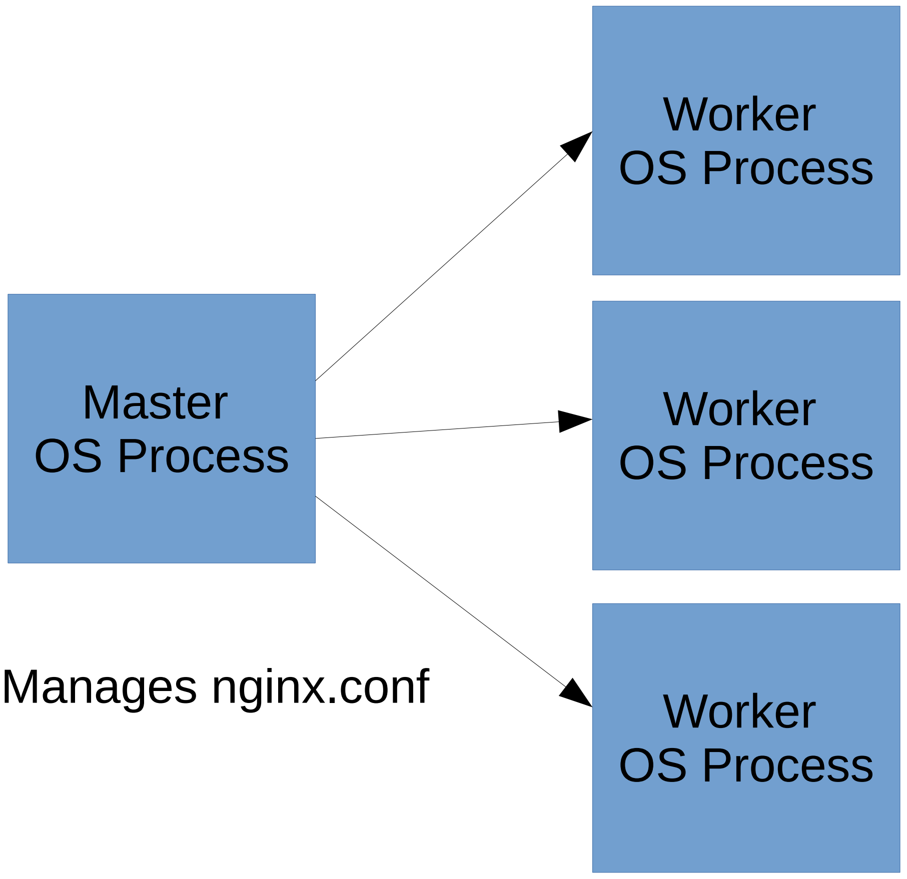

Browser
https
Load Balancer (AWS ELB, Nginx, etc.)
SSL Termination
http
Application Server (Python wsgi, Tomcat, etc.)
Security Perimeter such as - Internal/Private Network

Nginx

- Pronounciation
  - Engine-X

- https://www.nginx.com/resources/wiki/community/faq/

Nginx



Manages nginx.conf

https://docs.nginx.com/nginx/admin-guide/basic-functionality/runtime-control/

Nginx Load Balancing

- https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/

Nginx

```
http {
    upstream backend {
        server backend1.example.com weight=5;
        server backend2.example.com;
        server 192.0.0.1 backup;
    }

server {
        location / {
            proxy_pass http://backend;
        }
    }
}
```

Defined in nginx.conf

Requests coming at '/' will be redirected to one of the following servers: backend1.example.com, backend2.example.com, 192.0.0.1

AWS: Application Load Balancing

- https://aws.amazon.com/elasticloadbalancing/details/
- Layer-7 Load Balancing
  - i.e. HTTP/HTTPS traffic
- High Availability
  - Distribute incoming traffic to targets across multiple Available Zones
- Content-based Routing
  - Host-based Routing: Use the “Host” field in the HTTP header
  - Path-based Routing: Use the URL path in the HTTP header
- Containerized Application Support
  - Load balancing across multiple ports on a Single EC2 instance.
  - Deep Integration with Elastic Container Service (ECS)
    - ECS spins up new container on a random port and adds that container to the Load Balancer pool dynamically
- Logging
  - Load Balancer logs are stored in S3

GCP: Worldwide Load Balancing

- https://cloud.google.com/load-balancing/
- Global load balancing across regions
- Over 1 Million Queries per second
- Affinity
  - Direct and stick user traffic to specific backend instances

Kubernetes Load Balancing

Using Service Object with the 'LoadBalancer' type

```
 kind: Service
apiVersion: v1
metadata:
  name: greeting-service
spec:
   selector:
   app: greetings
 ports:
   - protocol: TCP
     port: 80
     targetPort: 5000
 type: LoadBalancer
```

A Load Balancer from underlying Cloud will be spun up when creating this Service Object

Reference

- https://www.upcloud.com/support/how-to-set-up-load-balancing/

Monitoring

Devdatta Kulkarni

What is Monitoring?

- Monitoring means to periodically check the state of a software entity
  - VM, Container, Application (through its url), etc.

- Why needed?
  - For seamless user experience
    - Think of Assignment 5 in which some of yours clusters are down and application URL (IP address) is unresponsive – Actively monitoring the cluster and taking actions when it is not responsive can reduce user perceived downtime
  - Make best utilization of compute resources
    - If VM is unresponsive but is still in 'Running' state then you are paying but not getting return for it!
  - To determine whether new instances of VMs/containers/application instances should be spun up or not
    - If traffic to the application has increased, add new application instances (VMs/containers)

AWS Monitoring: CloudWatch

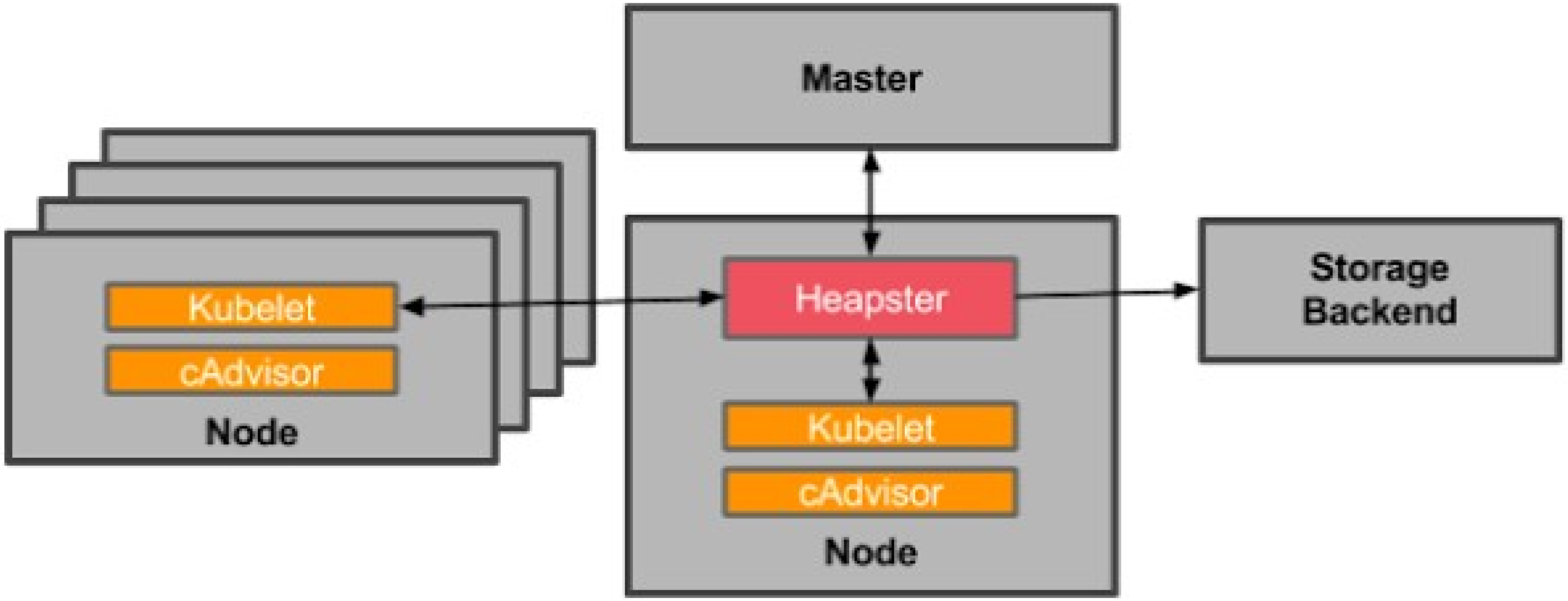
- https://aws.amazon.com/cloudwatch/
- Monitor AWS resources
  - EC2, DynamoDB tables, RDS DB instances, Elastic Load Balancers, etc.
- Monitor your applications
  - From the application, make calls to CloudWatch through its API
- Set Alarms
  - For example, when load on an application crosses some threshold, trigger creation of new application instance (auto-scaling)
- View Graphs & Statistics
- React to resource changes using CloudWatch events

GCP Monitoring: Stackdriver

- https://cloud.google.com/monitoring/
- Monitors resources from GCP and AWS
- Alerting
- Custom Metrics
- Dashboards

Kubernetes Monitoring: Heapster

https://kubernetes.io/docs/tasks/debug-application-cluster/resource-usage-monitoring/

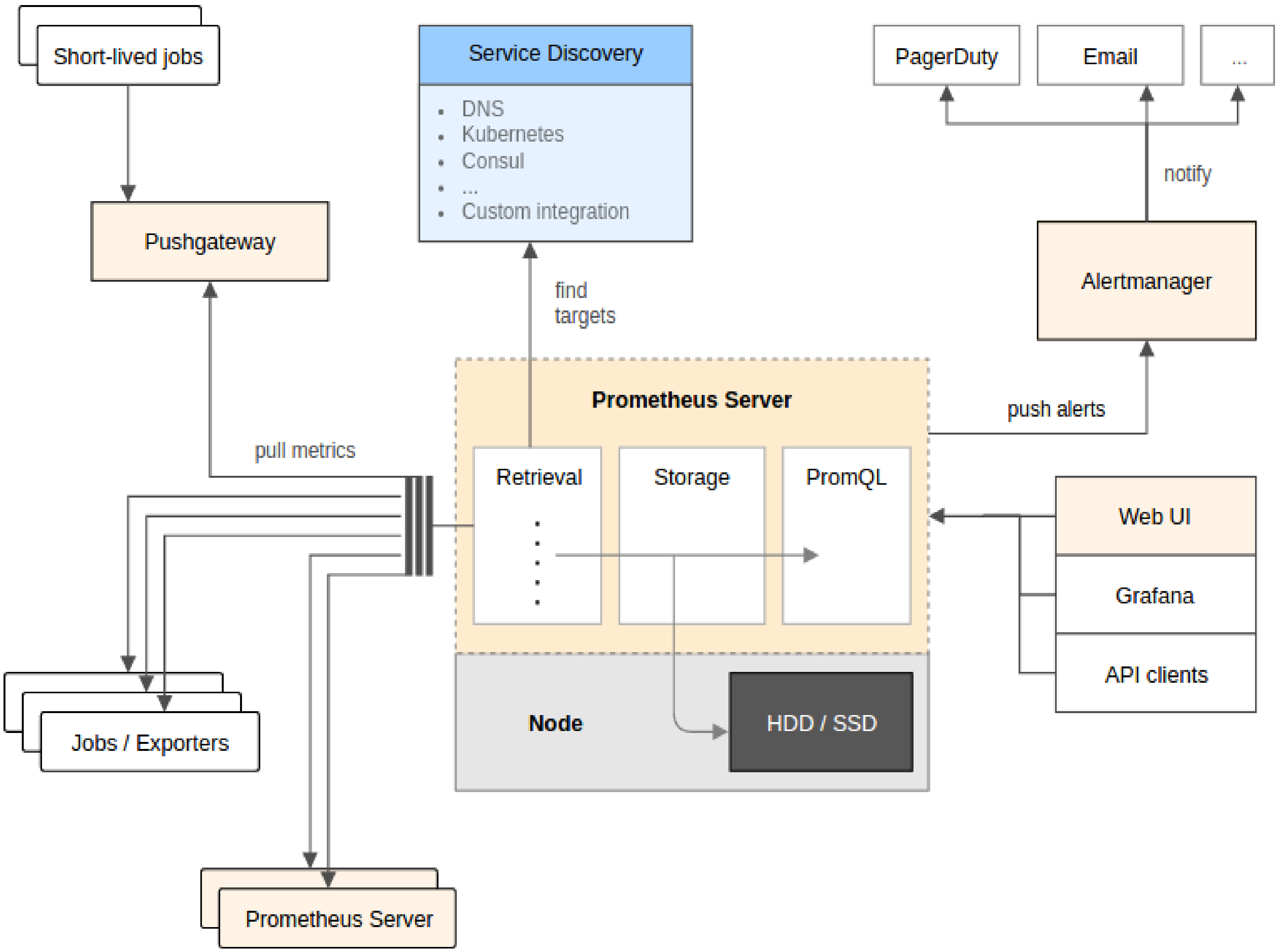


Kubernetes Monitoring: Prometheus

- https://prometheus.io/docs/introduction/overview/

- How does it work?
  - In your application, you implement generate required monitoring metrics (say number of requests received) and expose them at a well-known URL
  - Configure Prometheus to read from this application URL
  - Prometheus will periodically poll the URL and store data as a *timeseries*

Prometheus



Short-lived jobs
Pushgateway
Service Discovery
DNS
Kubernetes
Consul
...
Custom integration
find targets
PagerDuty
Email
...
notify
Alertmanager
push alerts
Prometheus Server
pull metrics
Retrieval
Storage
PromQL
Web UI
Grafana
API clients
Node
HDD / SSD
Jobs / Exporters
Prometheus Server

Monitoring done by Load Balancers

- Nginx
  - Has ability to monitor if a server is responsive or not
    - If it is not, Nginx will remove the server from load balancer pool
    - After some timeout period, Nginx will add the server back to the load balancer pool
- AWS Elastic Load Balancer
  - Homework question
- Google Network Load Balancer
  - Homework question

Cloud Applications

Devdatta Kulkarni

Cloud Applications

- AWS Whitepapers
  - Total Cost of Ownership of Web Applications
  - AWS Best Practices
- Cloud-Native Applications
  - Cloud-Native Computing Foundation
- Serverless Applications

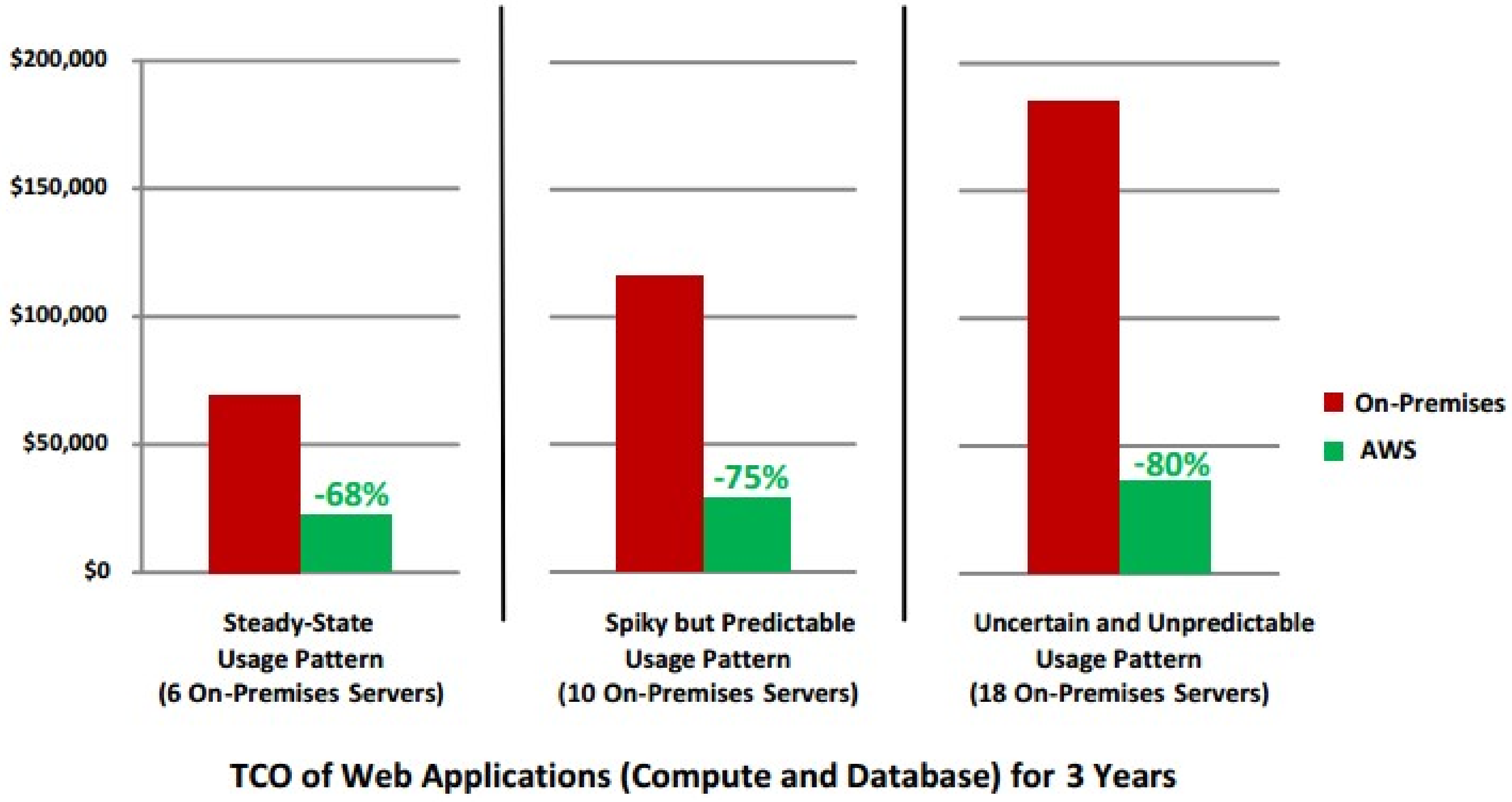
AWS Whitepapers

- The Total Cost of (Non) Ownership of Web Applications in the Cloud
- Architecting for the Cloud - Best practices
- AWS Well-Architected Framework

TC(N)O of Web Applications in the Cloud

- https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf
- Published in 2012
  - When people were not yet convinced about Cloud computing
- Traffic Patterns considered
  - Steady-state
  - Spiky but predictable
  - Uncertain and unpredictable

TCO summary



Ref: https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf

Usage Pattern: Steady-State

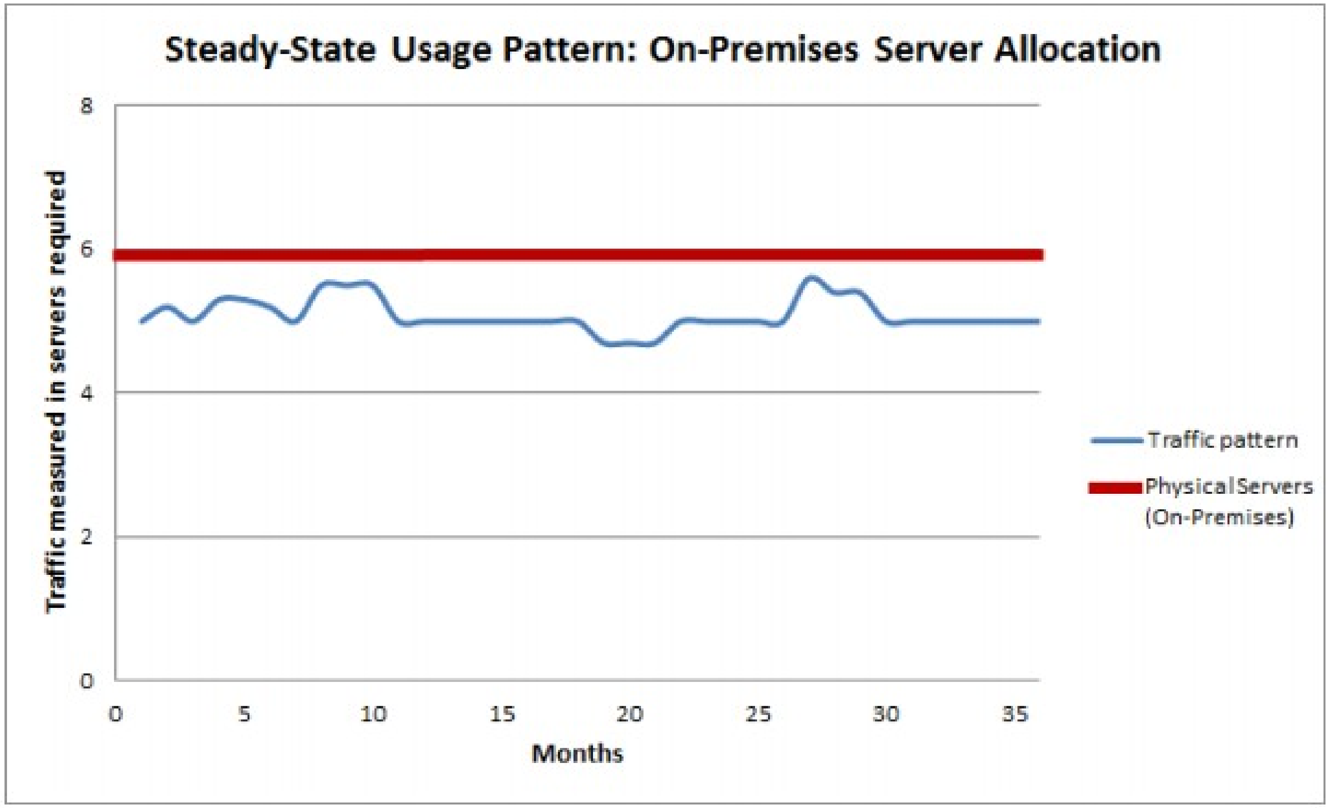
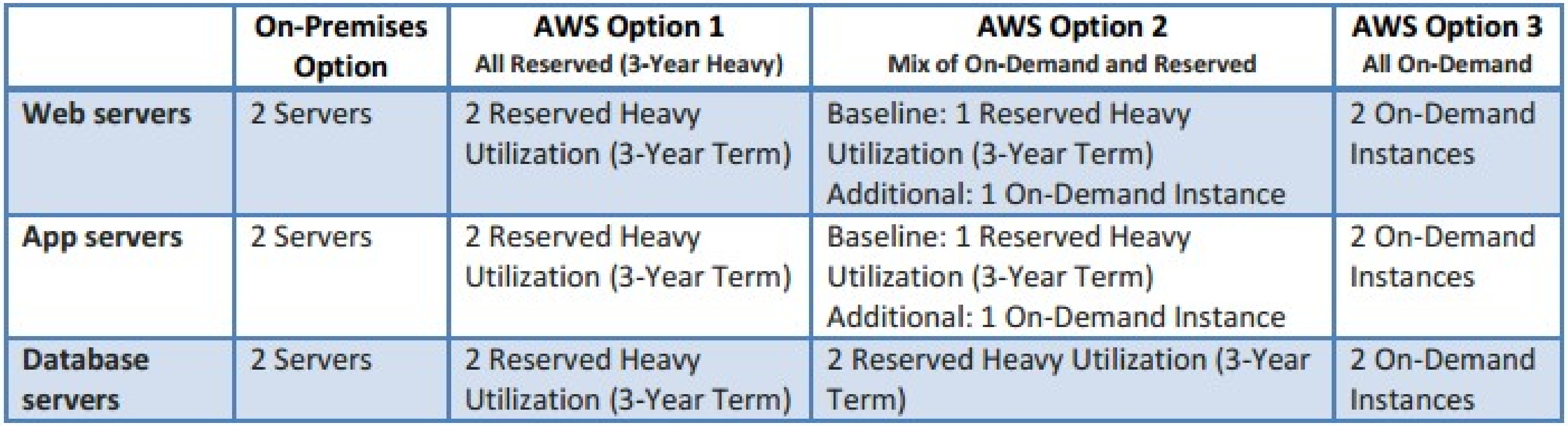


Figure 2: On-Premises Server Allocation for Steady-State Usage Pattern

Ref: https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf

Usage Pattern - Steady State

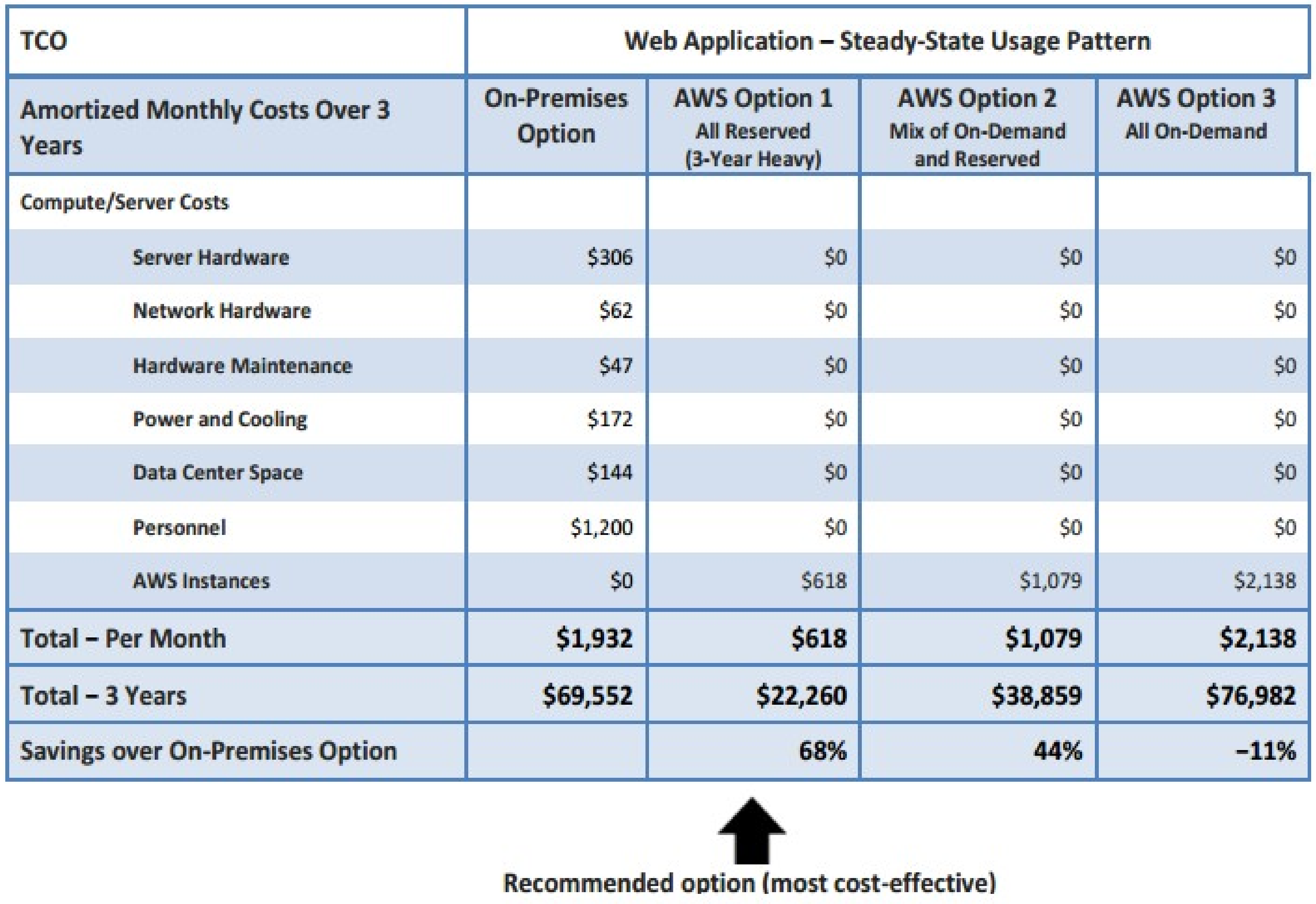


	On-Premises Option	AWS Option 1 All Reserved (3-Year Heavy)	AWS Option 2 Mix of On-Demand and Reserved	AWS Option 3 All On-Demand
Web servers	2 Servers	2 Reserved Heavy Utilization (3-Year Term)	Baseline: 1 Reserved Heavy Utilization (3-Year Term) Additional: 1 On-Demand Instance	2 On-Demand Instances
App servers	2 Servers	2 Reserved Heavy Utilization (3-Year Term)	Baseline: 1 Reserved Heavy Utilization (3-Year Term) Additional: 1 On-Demand Instance	2 On-Demand Instances
Database servers	2 Servers	2 Reserved Heavy Utilization (3-Year Term)	2 Reserved Heavy Utilization (3-Year Term)	2 On-Demand Instances

Table 3: Different Options Considered for Steady-State Web Application Scenario

Ref: https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf

Usage Pattern: Steady State



TCO	Web Application – Steady-State Usage Pattern			
Amortized Monthly Costs Over 3 Years	On-Premises Option	AWS Option 1 All Reserved (3-Year Heavy)	AWS Option 2 Mix of On-Demand and Reserved	AWS Option 3 All On-Demand
Compute/Server Costs				
Server Hardware	$306	$0	$0	$0
Network Hardware	$62	$0	$0	$0
Hardware Maintenance	$47	$0	$0	$0
Power and Cooling	$172	$0	$0	$0
Data Center Space	$144	$0	$0	$0
Personnel	$1,200	$0	$0	$0
AWS Instances	$0	$618	$1,079	$2,138
Total – Per Month	**$1,932**	**$618**	**$1,079**	**$2,138**
Total – 3 Years	**$69,552**	**$22,260**	**$38,859**	**$76,982**
Savings over On-Premises Option		**68%**	**44%**	**–11%**

Recommended option (most cost-effective)

Table 4: TCO Comparison – Steady-State Usage Pattern

Ref: https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf

Usage Pattern: Unpredictable

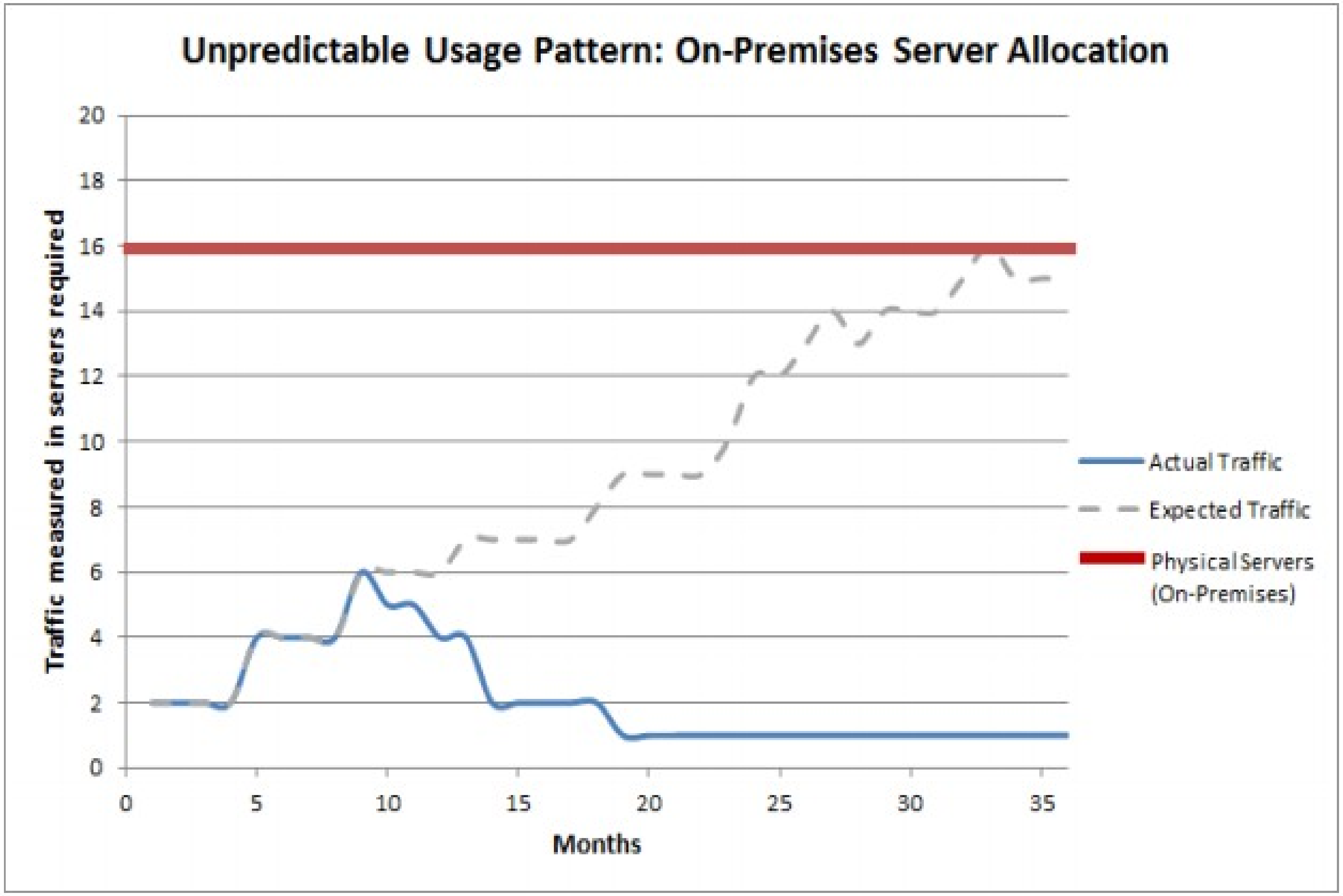
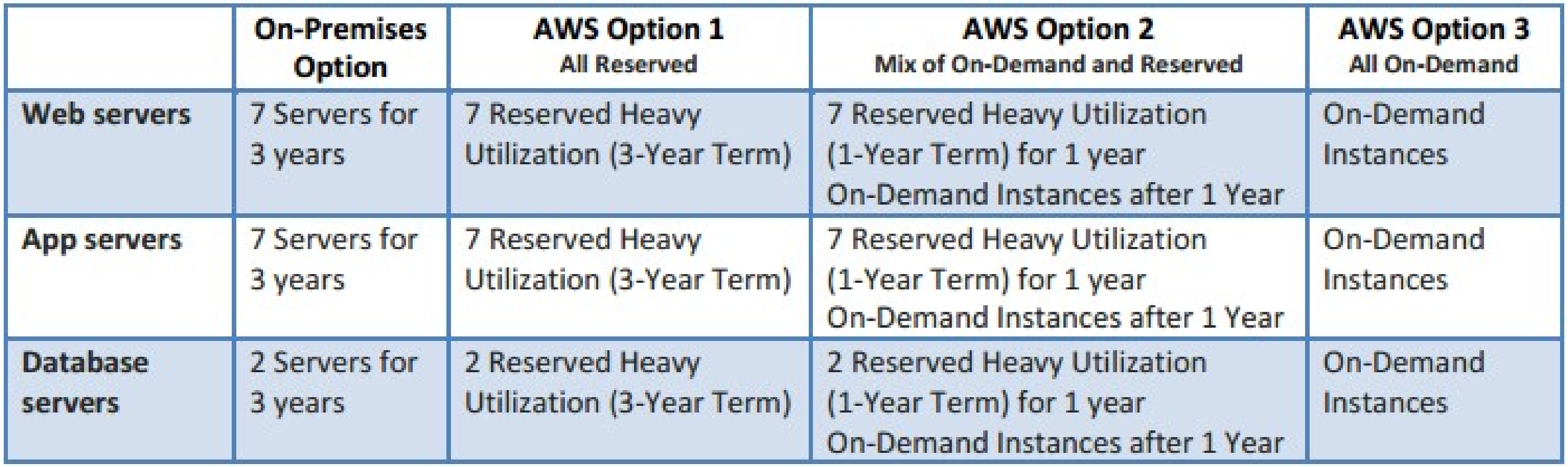


Figure 4: On-Premises Server Allocation for Uncertain and Unpredictable Usage Pattern

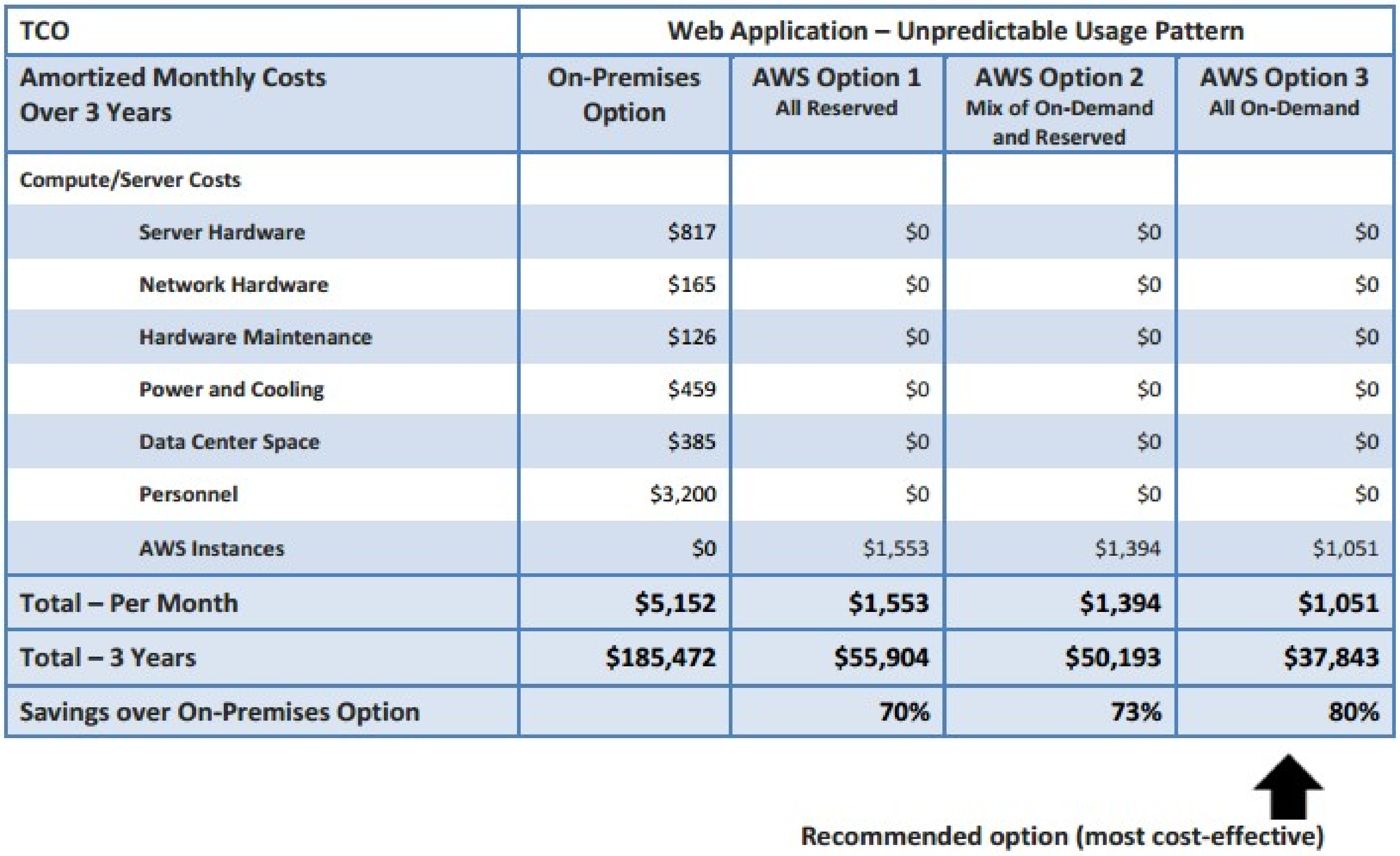
Usage Pattern: Unpredictable



	On-Premises Option	**AWS Option 1** All Reserved	**AWS Option 2** Mix of On-Demand and Reserved	**AWS Option 3** All On-Demand
Web servers	7 Servers for 3 years	7 Reserved Heavy Utilization (3-Year Term)	7 Reserved Heavy Utilization (1-Year Term) for 1 year On-Demand Instances after 1 Year	On-Demand Instances
App servers	7 Servers for 3 years	7 Reserved Heavy Utilization (3-Year Term)	7 Reserved Heavy Utilization (1-Year Term) for 1 year On-Demand Instances after 1 Year	On-Demand Instances
Database servers	2 Servers for 3 years	2 Reserved Heavy Utilization (3-Year Term)	2 Reserved Heavy Utilization (1-Year Term) for 1 year On-Demand Instances after 1 Year	On-Demand Instances

Table 8: Different Options Considered for Uncertain Unpredictable Web Application Scenario

Usage Pattern: Unpredictable



TCO	Web Application – Unpredictable Usage Pattern			
Amortized Monthly Costs Over 3 Years	**On-Premises Option**	**AWS Option 1** All Reserved	**AWS Option 2** Mix of On-Demand and Reserved	**AWS Option 3** All On-Demand
Compute/Server Costs				
Server Hardware	$817	$0	$0	$0
Network Hardware	$165	$0	$0	$0
Hardware Maintenance	$126	$0	$0	$0
Power and Cooling	$459	$0	$0	$0
Data Center Space	$385	$0	$0	$0
Personnel	$3,200	$0	$0	$0
AWS Instances	$0	$1,553	$1,394	$1,051
Total – Per Month	**$5,152**	**$1,553**	**$1,394**	**$1,051**
Total – 3 Years	**$185,472**	**$55,904**	**$50,193**	**$37,843**
Savings over On-Premises Option		**70%**	**73%**	**80%**

Recommended option (most cost-effective)

Table 9: TCO Comparison – Unpredictable Usage Pattern

Companies Interviewed

- Airport Nuremberg
- Foursquare Labs
- Global Blue
- Hitachi
- Junta de Andalucia
- NASDAQ
- NASA/JPL
- Newsweek
- Pfizer
- Razorfish
- Samsung

Architecting for the Cloud

- https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf

- Best Practices of designing Applications for the Cloud
  - Published in 2016
  - When people were asking “What are the best practices to design applications for Cloud”

Architecting for the Cloud

- Scalability
  - Horizontal vs. Vertical
- Removing Single Points of Failure
  - Redundancy, Sharding (requests splitting), multi-region deployments
- Optimize for Cost
  - Reserved capacity, Spot instances
- Caching
  - Edge caching for static content
- Security
  - Reduce Privileged Access, Audits
- Disposable Resources Instead of Fixed Servers
- Automation
  - PaaS, Cloud Formation
- Loose Coupling
  - Multi-layer architectures with Queues
- Managed Services, Not Servers

AWS Well-Architected Framework

- https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

- Date Published: November 2017

AWS Well-Architected Framework

- Five aspects (as defined by AWS)
  - Operational Excellence
  - Security
  - Reliability
  - Performance Efficiency
  - Cost Optimization

Operational Excellence

- Perform operations as code
- Automatic generation of documentation
- Frequent, small, reversible changes
- Refine operations procedures frequently
- Anticipate failures
  - Perform “pre-mortem” exercises

Security

- Implement a strong identity foundation
  - IAM
- Enable traceability
  - Monitor, alert, audit actions and changes to your environment in real time
- Apply security at all layers
  - VPC, subnet, load balancer, Multi-factor auth, physical data center security
- Automate security best practices
  - Security Policy as code
- Protect data in transit and at rest
  - Encryption
- Prepare for security events

Reliability

- Test recovery procedures
  - Simulate different failures or recreate scenarios that led to failures before
- Automatically recover from failure
  - Monitoring of Key Performance Indicators (KPIs) and triggering actions if threshold is crossed
- Scale horizontally to increase aggregate system availability
  - Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall system
  - Distribute requests to avoid common point of failure
- Stop guessing capacity
  - Monitor demand and system utilization to perform autoscaling
- Infrastructure-as-Code
  - Changes to the infrastructure should be done using automation

Performance Efficiency

- Use Managed Services
  - Relational/Non-relational data stores, Container Orchestration Engines, Machine Learning, etc.
- Use Multi-region deployments
- Use appropriate architectures as per the application use-case
  - Monolithic, Microservice, Serverless

Cost Optimization

- Pay-as-you-use
  - Increase/Decrease your infrastructure footprint as needed
- Analyze and attribute expenditure
- Use managed services to reduce cost of ownership

Cloud-Native Applications

- https://www.cncf.io/about/faq/

- What are Cloud-Native Applications?
  - They are containerized - each part is packaged in its own container
  - Dynamically orchestrated - Containers are actively scheduled and managed to optimize resource utilization
  - Microservices oriented - Applications are segmented into microservices

Cloud-Native Computing Foundation

- https://www.cncf.io/
- Formed: 2016
- Kubernetes is hosted by CNCF

- CNCF Conference (kubeconf 2018) happening this week in Denmark
  - https://events.linuxfoundation.org/events/kubecon-cloudnativecon-europe-2018/
  - May 2 - 4 2018

Serverless Applications

- https://d1.awsstatic.com/whitepapers/serverless-architectures-with-aws-lambda.pdf

- What are Serverless Applications?
  - Deploy applications without the need to provision and manage any servers
  - Business logic is deployed in the form of “functions” that are executed based on a trigger event

AWS Lambda

- AWS Lambda functions can be triggered

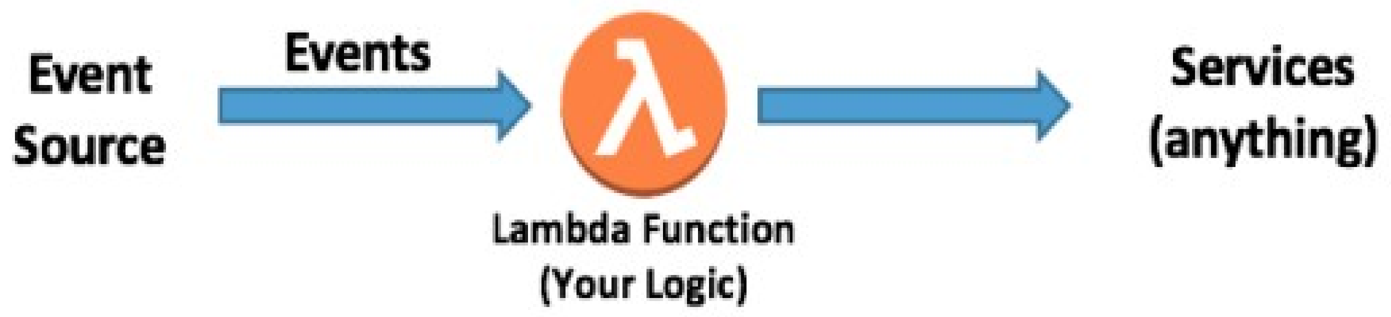
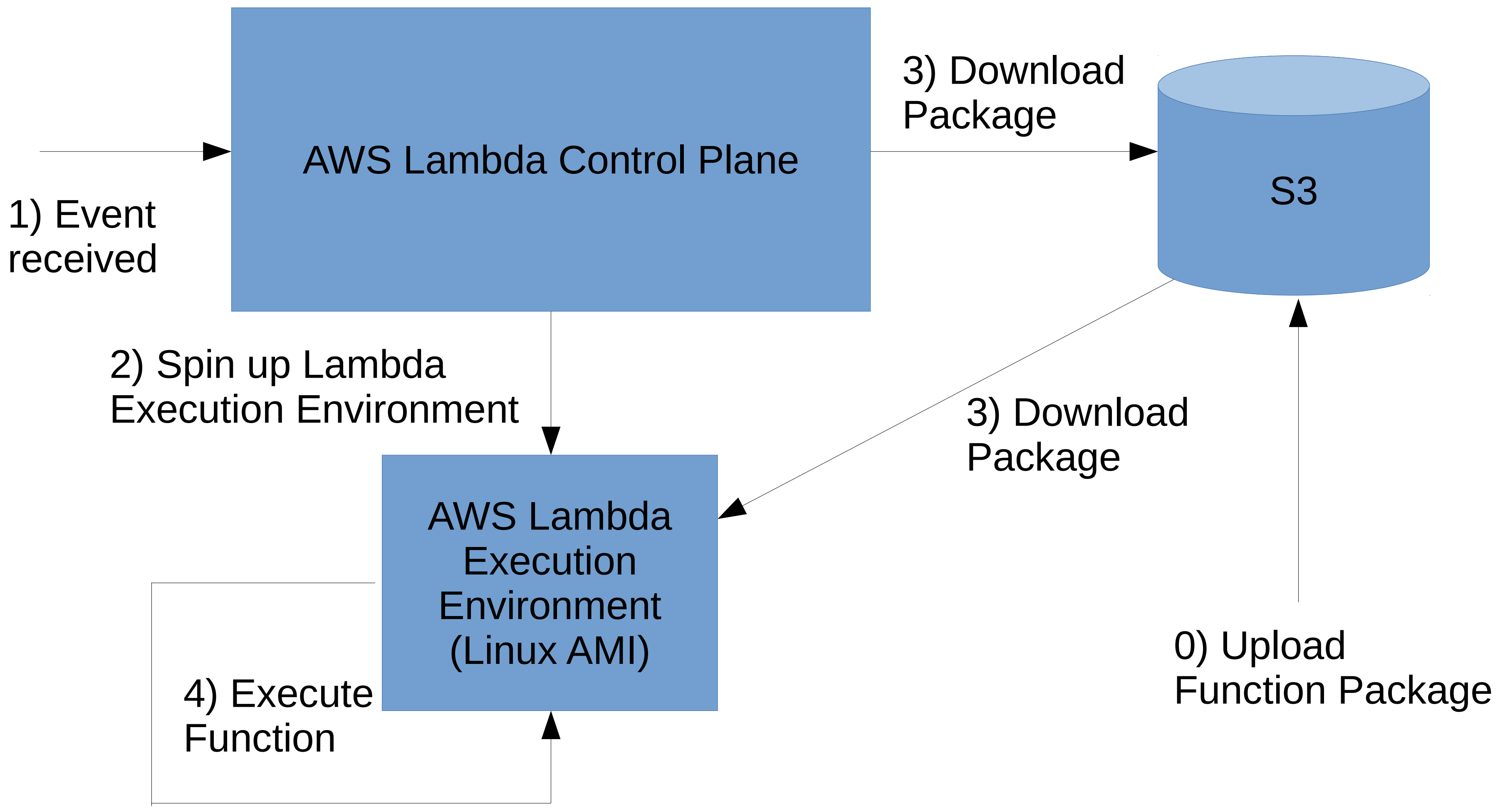


Figure 2: Simplified architecture of a running Lambda function

AWS Lambda Control Flow



AWS Lambda Control Plane
1) Event received
3) Download Package
S3
2) Spin up Lambda Execution Environment
3) Download Package
AWS Lambda Execution Environment (Linux AMI)
4) Execute Function
0) Upload Function Package

AWS Lambda: Handler

- The code to be executed is defined inside a “handler” function:

```
def handler_name(event, context):
 ...
 return some_value
```

AWS Lambda: Handler

- The code to be executed is defined inside a “handler” function:

```
def my_handler(event, context):
    message = 'Hello {} {}!'.format(event['first_name'],
                                    event['last_name'])
    return {
        'message' : message
    }
```

AWS Lambda: Event Object

- Contents of the event parameter include all the data and metadata required for your function's logic
  - Event created by API Gateway contains:
    - HTTPS request body, path, query string
  - Event created by S3 when a new object is inserted contains:
    - Details about the Bucket and the new object

AWS Lambda: Context Object

- Context Object allows Lambda function to interact with the Lambda execution environment
  - AWS RequestId
  - Remaining time:
    - Time in milliseconds before function timeout occurs
    - A Lambda function can execute for 300 seconds (5 minutes) before it is timed out

Configuration Options

- Function memory (RAM)
  - 128 MB minimum
  - 1.5 GB maximum
- Function versioning
  - $LATEST
- IAM Roles
  - Each Function needs to be granted a IAM role with required policies (this roles is called 'execution role')
- Environment variables support available
  - 12-factor - separate configuration from code factor is in play!

Case Studies

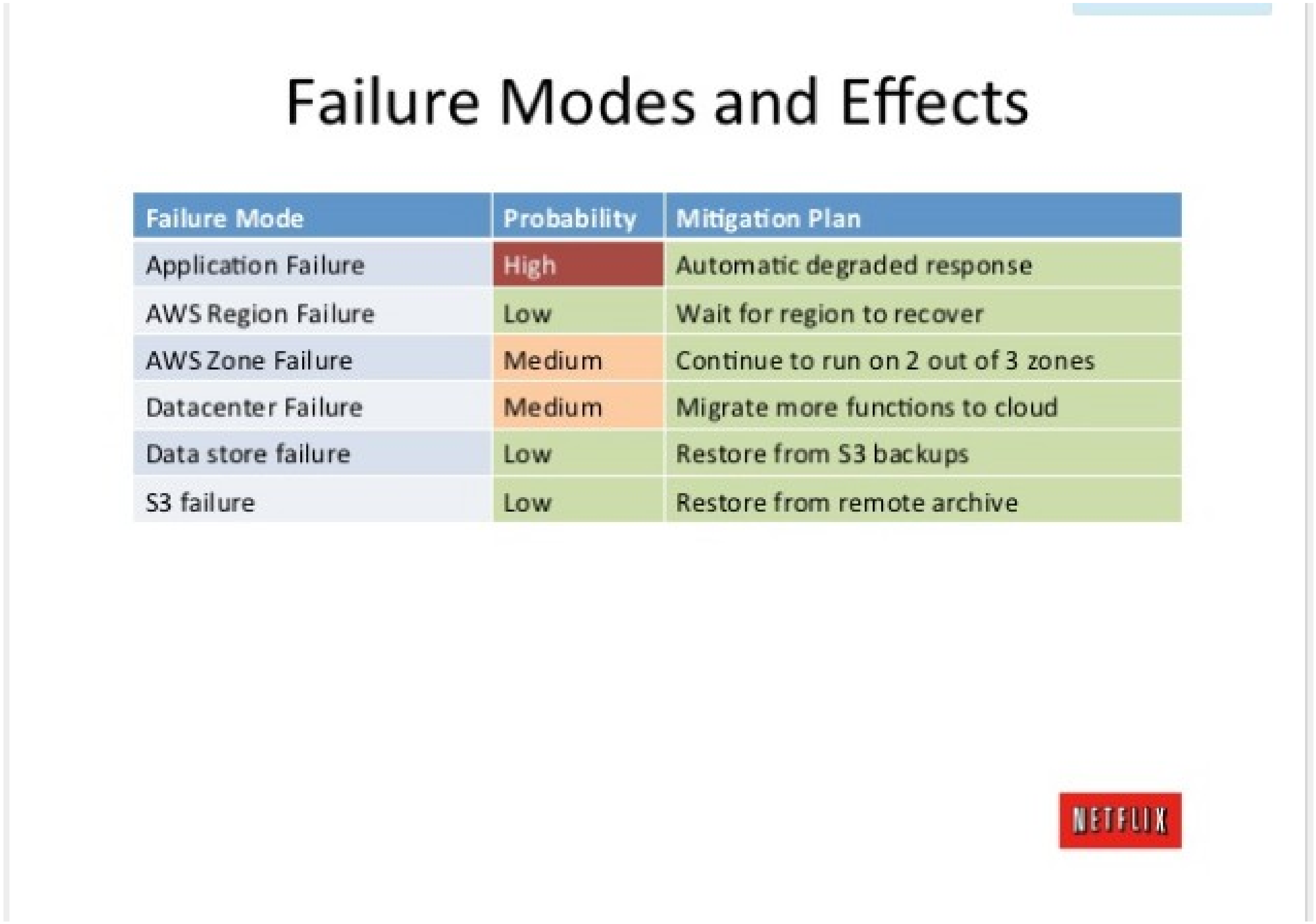
Devdatta Kulkarni

Netflix

- https://www.slideshare.net/adrianco/netflix-global-cloud
  - Published in 2012

- Highlights:
  - Use several AWS Services
  - Each Service is deployed in Three Balanced Availability Zones
  - Persistent storage is triple replicated
  - Services in different regions don't talk to each other
    - US and EU services don't talk to each other

Netflix Failure Modes and Mitigation

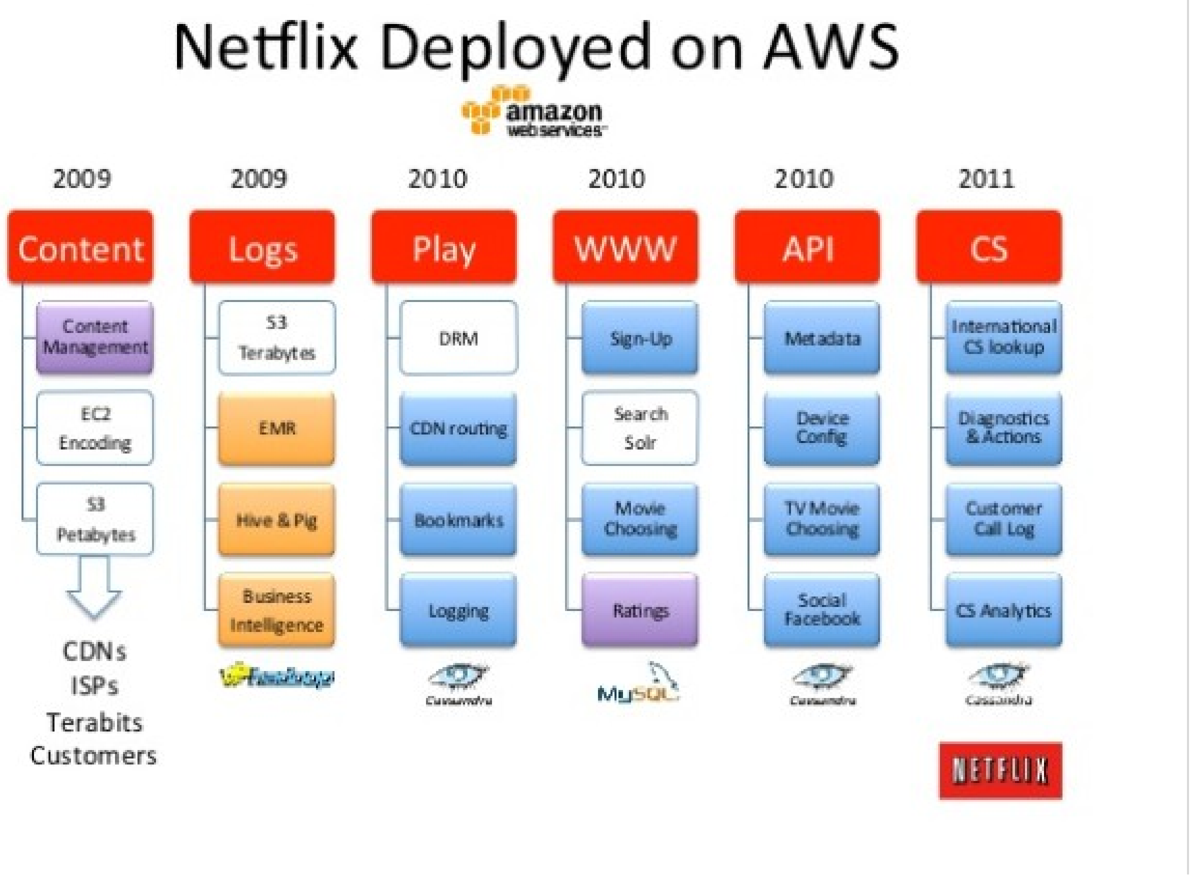


Failure Modes and Effects

Failure Mode	Probability	Mitigation Plan
Application Failure	High	Automatic degraded response
AWS Region Failure	Low	Wait for region to recover
AWS Zone Failure	Medium	Continue to run on 2 out of 3 zones
Datacenter Failure	Medium	Migrate more functions to cloud
Data store failure	Low	Restore from S3 backups
S3 failure	Low	Restore from remote archive

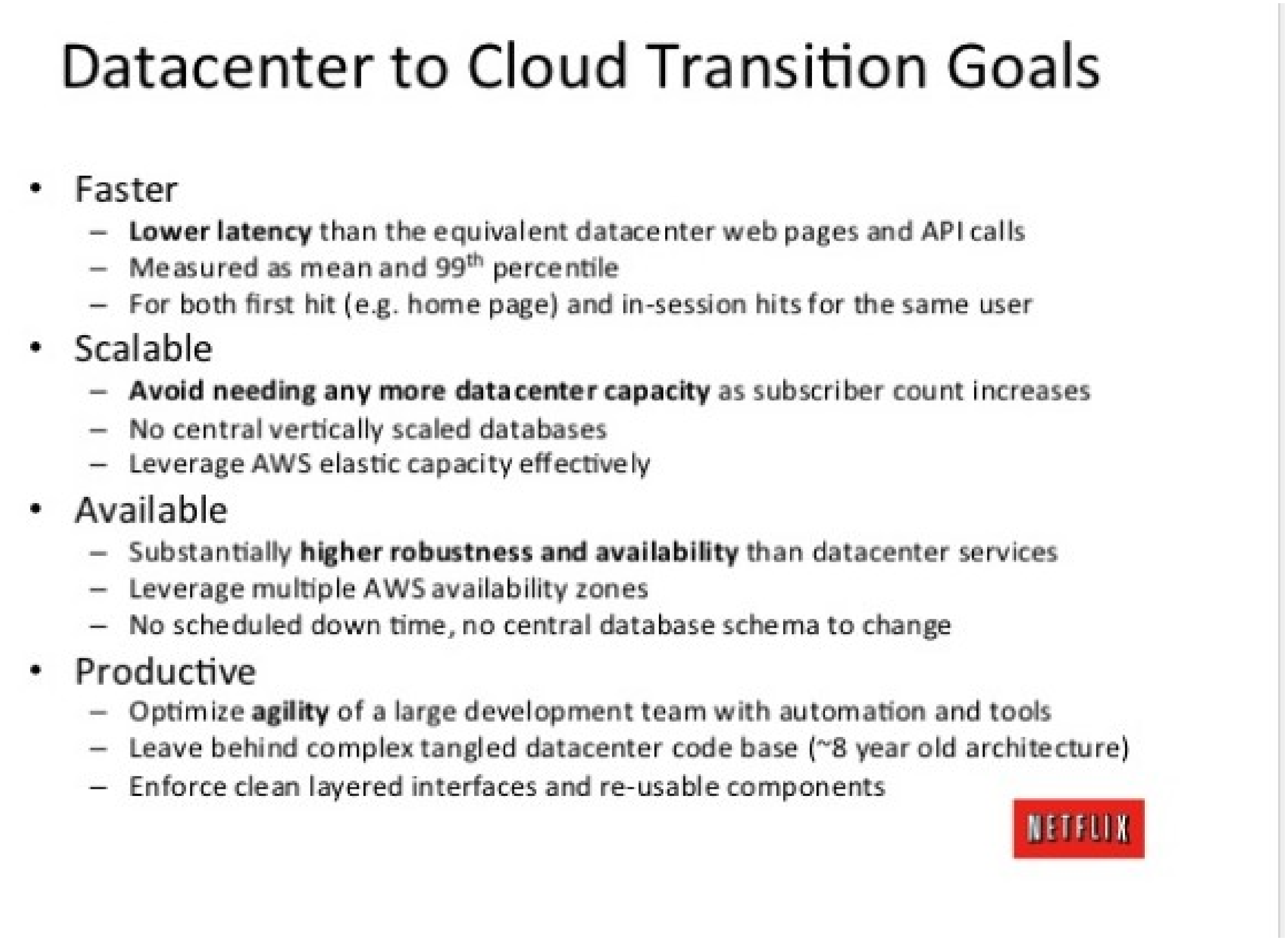
Reference: https://www.slideshare.net/adrianco/netflix-global-cloud

Netflix on AWS



Reference: https://www.slideshare.net/adrianco/netflix-global-cloud

Netflix on AWS



Datacenter to Cloud Transition Goals

- Faster
  - **Lower latency** than the equivalent datacenter web pages and API calls
  - Measured as mean and 99th percentile
  - For both first hit (e.g. home page) and in-session hits for the same user
- Scalable
  - **Avoid needing any more datacenter capacity** as subscriber count increases
  - No central vertically scaled databases
  - Leverage AWS elastic capacity effectively
- Available
  - Substantially **higher robustness and availability** than datacenter services
  - Leverage multiple AWS availability zones
  - No scheduled down time, no central database schema to change
- Productive
  - Optimize **agility** of a large development team with automation and tools
  - Leave behind complex tangled datacenter code base (~8 year old architecture)
  - Enforce clean layered interfaces and re-usable components

NETFLIX

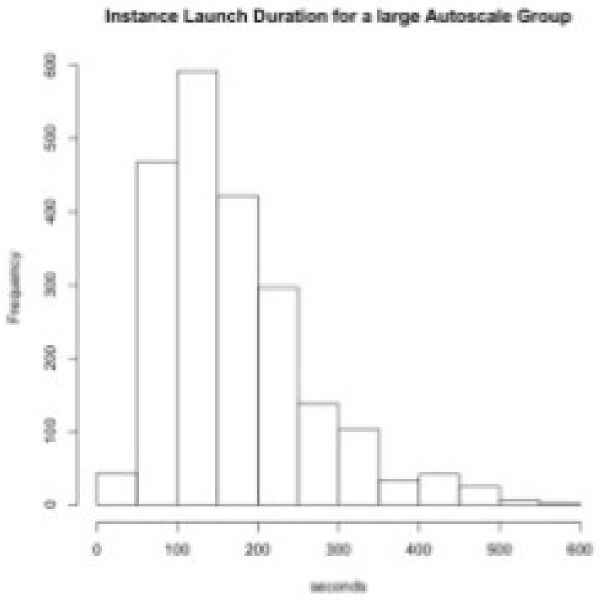
Reference: https://www.slideshare.net/adrianco/netflix-global-cloud

Netflix on AWS

Cloud Deployment Scalability

New Autoscaled AMI – zero to 500 instances from 21:38:52 - 21:46:32, 7m40s
Scaled up and down over a few days, total 2176 instance launches, m2.2xlarge (4 core 34GB)

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
41.0   104.2   149.0   171.8   215.8   562.0
```



Reference: https://www.slideshare.net/adrianco/netflix-global-cloud

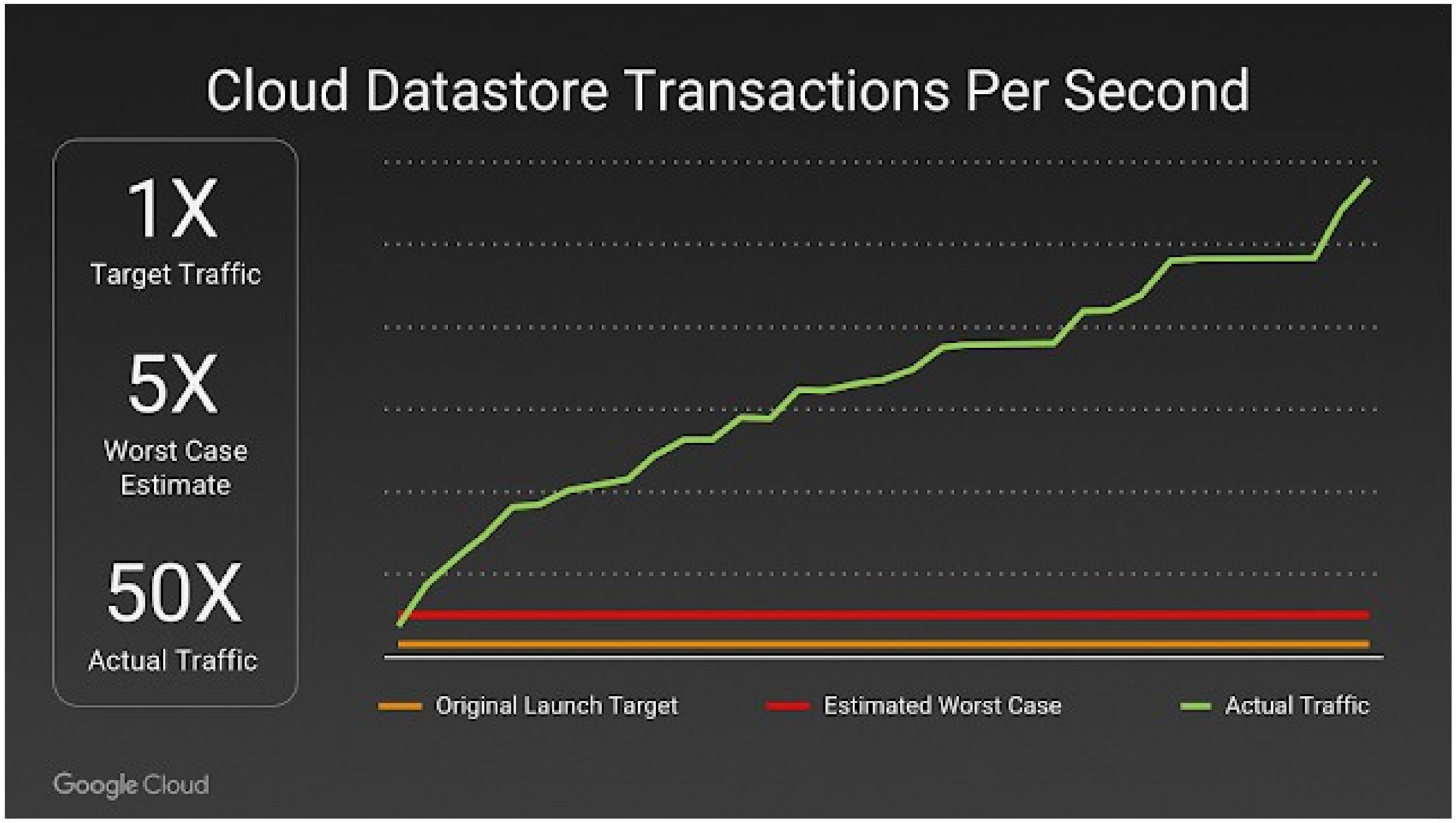
Netflix today: Container Platform

- Announced Recently (April 18, 2018)
  - https://medium.com/netflix-techblog/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436
- Containers running on EC2 instances
- Key features being used:
  - EC2 VPC API rate limits
  - EC2 metadata proxy
  - Application Load Balancers
- Half-million containers and 200,000 clusters per day!!
- https://www.slideshare.net/aspyker/container-world-2018

Niantic

- https://cloudplatform.googleblog.com/2016/09/bringing-Pokemon-GO-to-life-on-Google-Cloud.html
- Published: November 2016
- Creator of Pokemon GO

Niantic on GCP



https://cloudplatform.googleblog.com/2016/09/bringing-Pokemon-GO-to-life-on-Google-Cloud.html

Pokemon GO on GCP

- Application logic runs in containers on GKE
  - GKE chosen because
    - Ability to orchestrate containers across all the GCP regions (“planetary-scale”)
- Google Cloud Datastore is used for Backend storage
  - https://cloud.google.com/datastore/docs/
  - It is a NoSQL database (similar to DynamoDB)
- HTTP/S Load Balancer is used over Network Load Balancer
  - https://cloud.google.com/load-balancing/
  - Worldwide autoscaling and load balancing
- In total: Over a dozen GCP services used
- Largest GKE deployment till now!

Pokemon GO on GCP

- Live upgrade of GKE cluster with more than thousand additional nodes to prepare for launch in Japan
  - Requirement: No downtime for existing players