Clouds

CS199 - ACC

Prof. Robert J. Brunner

Ben Congdon
Tyler Kim
Bhuvan Venkatesh

Clouds

- **“Private” Clouds**
  - Used for a company’s internal services only
  - Example: Internal datacenters of companies like Facebook, Google, etc.

- **“Public” Clouds**
  - Anyone can purchase resources
  - You can build your own company on top of another company’s cloud
  - Example: AWS, GCP, Azure

Why use a cloud?

- **Reliability**
  - It's someone else's responsibility to fix broken machines
- Cheap and On-Demand **Scalability**
  - Pricing is per hour or second instead of sunk hardware cost
  - Can create and destroy nodes on a *per second* basis
- **Hardware** Abstraction
  - Don't have to care about underlying hardware, just the specs of your VM
- "Special Sauce"
  - Proprietary features (i.e. AWS DynamoDB or Google BigQuery)

Cloud Providers

The Giants

Google Cloud Platform

Amazon Web Services (AWS)

- The largest by far of the public clouds
  - You use it every day and don't even know it
  - Netflix, Reddit, Spotify, and millions others
- When it goes down, the half of the internet goes down
  - Example: The infamous S3 outage in February 2017

AWS Offerings

Compute
EC2
EC2 Container Service
Lightsail
Elastic Beanstalk
Lambda
Batch

Storage
S3
EFS
Glacier
Storage Gateway

Database
RDS
DynamoDB
ElastiCache
Redshift

Networking & Content Delivery
VPC
CloudFront
Direct Connect
Route 53

Migration
Application Discovery Service
DMS
Server Migration
Snowball

Developer Tools
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
X-Ray

Management Tools
CloudWatch
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Trusted Advisor
Managed Services

Security, Identity & Compliance
IAM
Inspector
Certificate Manager
Directory Service
WAF & Shield
Compliance Reports

Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
Data Pipeline
QuickSight

Artificial Intelligence
Lex
Polly
Rekognition
Machine Learning

Internet Of Things
AWS IoT

Contact Center
Amazon Connect

Game Development
Amazon GameLift

Mobile Services
Mobile Hub
Cognito
Device Farm
Mobile Analytics
Pinpoint

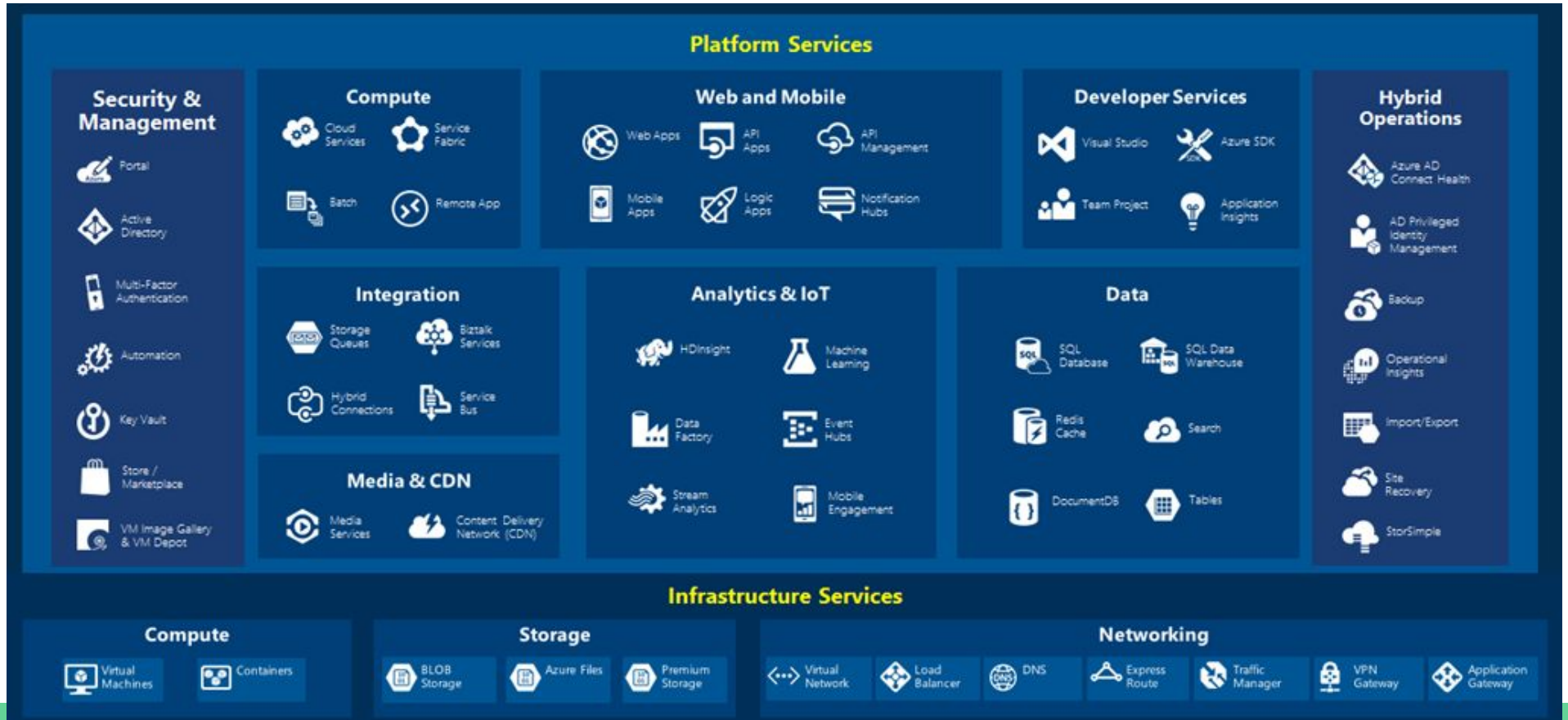
Application Services
Step Functions
SWF
API Gateway
Elastic Transcoder

Messaging
Simple Queue Service
Simple Notification Service
SES

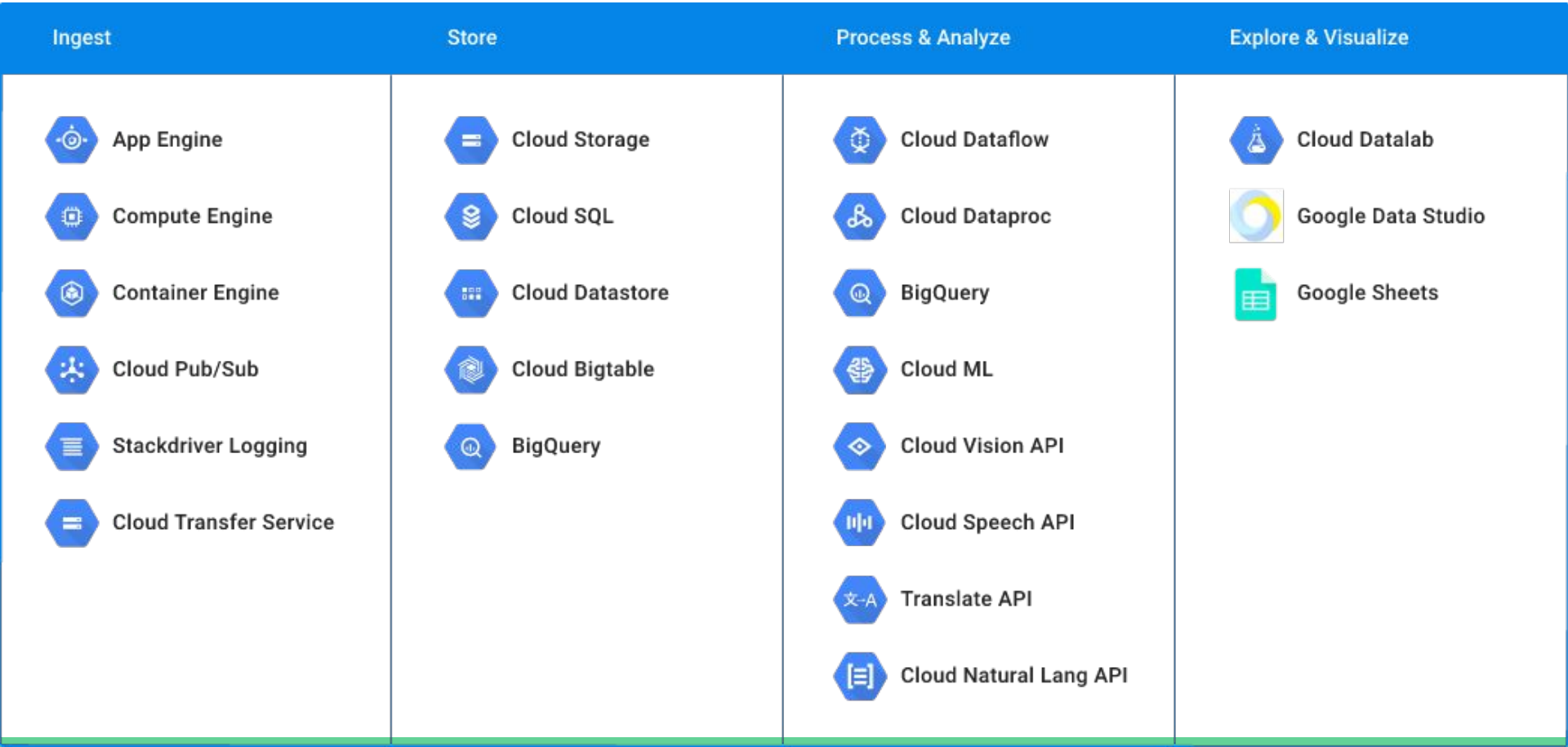
Business Productivity
WorkDocs
WorkMail
Amazon Chime

Desktop & App Streaming
WorkSpaces
AppStream 2.0

Azure Services



Google Cloud Platform



Feature Parity

- All clouds try to compete on features so they all end up having extremely similar feature sets

Virtual Machines

AWS Elastic Compute Cloud (EC2)

- The basic one which all of these clouds provide are Virtual Machines
- AWS has everything from the tiny to gigantic monsters
  - T2.Nano: 1 VCPU 512 MB Ram
  - X1.32xlarge: 128 VCPU 2000 GB Ram (One of these is more powerful than our cluster)
- They have GPUS!
  - Can do deep learning
- Most are fixed price per hour but there is a price auction for unused machines
  - Lets you do stuff super cheap as long as your program can handle getting a shutdown notice within 30 seconds

Azure Virtual Machines

- Similar to AWS
- GPUs
- Not as many CPUs (Max is 32 currently)
- Not as much ram (Max 800 GB currently)
- But you probably will not hit these limits

Google Compute Engine

- Provides VMs
- Largest server is 64 VCPU, 416 GB Ram
- Provides custom sized machines
- Cost is per minute!!

Storage

Storage

- AWS Simple Storage Service (AWS S3)
  - Massive storage, a ton of the internet stores all their content here.
    - Imgur
- Google Cloud Storage
- Azure Storage

Hosted Data Processing

- Hosted Hadoop, Spark, HBase, Presto, Hive clusters
- Does all the management for you
- Is extremely reliable (more than our current cluster sadly)

- Amazon Elastic Map Reduce
- Microsoft HDinsight
- Google Dataproc

Databases

- Let the clouds manage your database hosting
  - Does create tables and stuff for you, just the stuff below it
- AWS
  - DyanamoDB
  - Relational Database Server (RDS)
- GCP
  - BigTable
  - BigQuery
  - CloudSQL
  - Spanner
- Azure
  - MSSQL
  - DocumentDB

Unique Features

- GCP
  - CloudSpanner
    - A planet distributed database
    - CP System
  - Tensor Processing Unit
    - Do deep learning in hardware
- AWS
  - Absurdly large feature set
  - FPGAs
- Azure

Infrastructure as Code

(A.K.A. How to train your Cloud)

How do we setup our Cloud Applications?

Approaches:

- **Setup everything manually!**
  - Does this scale? Clearly no.
- **Custom scripts**
  - Use your cloud provider's API to create machines
  - Programatically SSH into the machine to do tasks
  - Does this scale? Maybe... but why reinvent the wheel?
- **Infrastructure as Code**
  - Declare your infrastructure setup in a specific format
  - Your IaC framework deploys/updates your cloud infrastructure!
  - Does this scale? Yes!

Infrastructure as Code Ideas

- Approaches to “writing down” cloud configuration:
    - **Declarative:** Define the target state of your cloud. *What* should the eventual cloud deployment look like?
    - **Imperative:** Define how the configuration system should setup the cloud. *How* should the system deploy your application?
    - **Intelligent:** Define relationships and constraints between services, and the system will figure out *how* and *what* to update.

Infrastructure as Code Ideas

- Approaches to updating cloud configuration:
    - **Push**: A central server tells child servers their configuration
    - **Pull**: Child servers request configuration from a central server

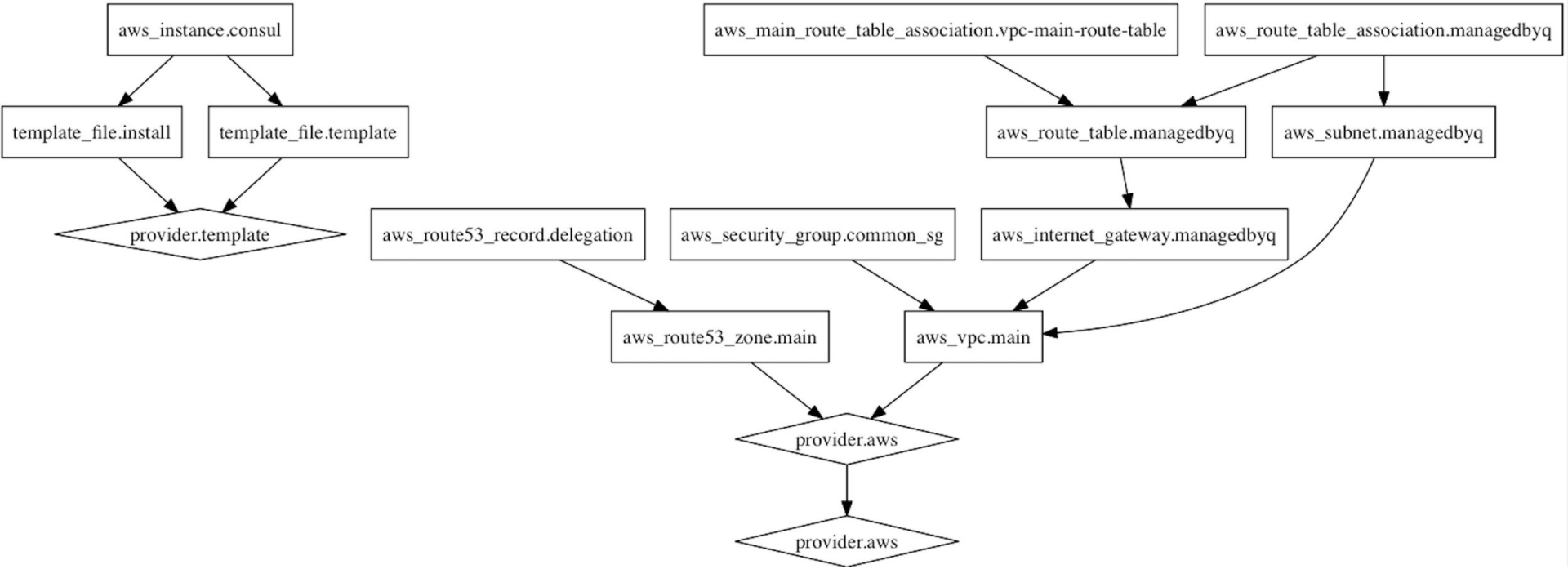
Infrastructure as Code Solutions

- **Ansible:** Declarative/Imperative; Push
- **Puppet:** Declarative; Pull
- **Chef:** Imperative; Pull
- **Salt**: Declarative
- **Terraform:** Declarative/Intelligent; Push

Terraform

- Created by HashiCorps; Open source
- Cloud Platform Agnostic
  - Support for AWS, GCP, Azure, Kubernetes, Heroku, and a bunch more
- Stateful and environment aware
  - Internal resource graph used to create cloud resources in the correct order

Terraform Resource Graph



https://github.com/hashicorp/terraform/issues/1997

Terraform Modules

- Terraform uses *.tf files for configuration
- Common semantics:
  - variables.tf
    - Hold variables that may change over the lifetime of the configuration
    - i.e. Instance sizing, database table names, etc.
  - main.tf
    - Import variables and any necessary modules.
  - Others (i.e. ec2.tf)
    - Service-specific configuration
    - Usually 1-file-per-service (i.e. one for EC2, and another for DynamoDB)

Terraform Syntax

- Can be rather confusing.
- Basic “language” that supports some interpolation, but is generally declarative
- Useful to lookup and use examples
  - Many open-source Terraform templates are available

Terraform Commands

- terraform get
  - Downloads and updates local terraform modules
- terraform plan
  - Creates an execution plan to transform the state in your cloud to the state of your current local configuration
- terraform apply
  - Runs the execution plan, and creates/updates/deletes resources in your cloud as necessary
  - Can be a destructive action if you're not careful!