**Overview:**

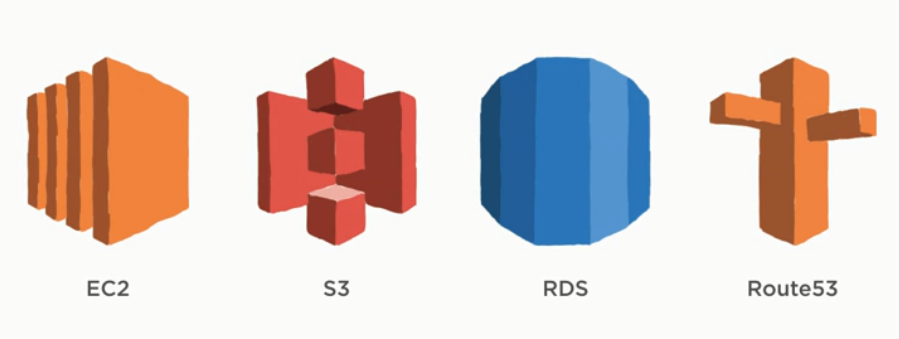
* **Cloud:**

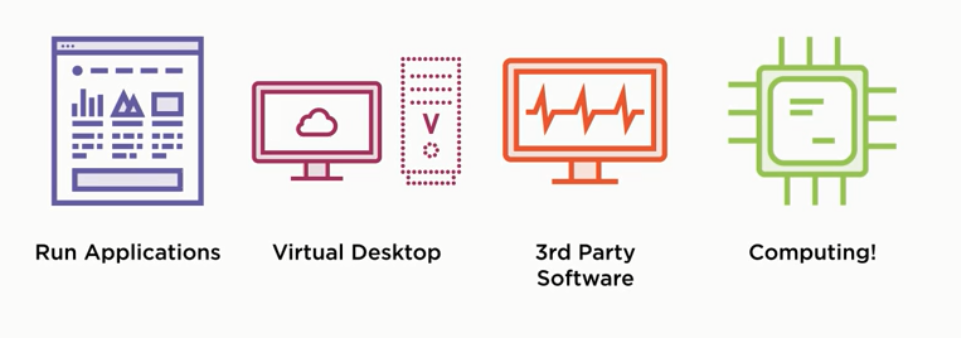
Cloud services in general like AWS and Azure solve the overhead and complexity of the on-primes severs.

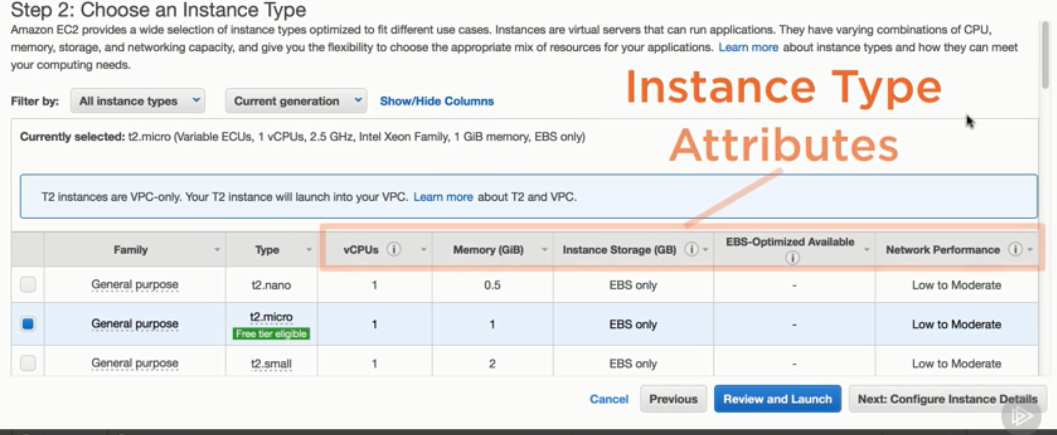
* **Why AWS?**
  + AWS is a collection of cloud computing services that can be utilized individually or in tandem to run a computer application. To break this down, we call it cloud computing services because the services are running in some magical data center somewhere, and marketing people need simple terms for these things. These services all provide some type of computing operation, whether it be literal computing, messaging, storage, routing, or something else. Although AWS is typically used for web applications, that is not a requirement. You can run any type of application with AWS. Many universities, government agencies, and corporations use the flexibility of AWS to process large amounts of data or run a lot of computations that would be difficult and expensive to set up on-premises. Amazon Web Services is simply one of the many providers of cloud services. They have distinguished themselves as one of the most reliable and also as having the largest geographical footprint. Large tech enterprises have their entire infrastructure running on AWS, which is a huge vote of confidence for the service. Essentially, if AWS went down, there would be no Netflix.
  + AWS and other providers provide to you the cost control (pay as you go for only what you use) and scalability.
  + AWS have a calculator like in Azure to calculate the cost.
  + It’s too easy in AWS to deploy your app or service globally around the world for reducing latency (وقت الاستجابة) and adding redundancy (وفرة)
  + AWS provide a AWS service health dashboard: <https://status.aws.amazon.com/>
* **How does AWS work:**
  + AWS is a collection of services that interact together to enable an application to run. In high level, Nearly all services interact with each other over typical TCP connections. It really depends on the service as to which port it might use. For instance, each type of database you might be using has its own port of choice, so there's not a single protocol that all services use.
  + Usually, you are initiating these connections in your application by making requests to the AWS services. If you create all of your service instance in the same virtual private cloud, then they will have local IP addresses, and you can make superfast connections to them.
  + When beginning your development with the various services of AWS, it's not an all or nothing process. Since all the services interact mostly the same on your local development machine as they do in they do in the cloud, you can add and build on services as you like. For example, a common first step for a content management system web application might be to connect to a database service in AWS. Then you could connect and begin using S3 for static file storage. You can work with these services exactly the same on your development environment as you would in the cloud, making the development to deployment experience as painless as possible.
  + The biggest change once you move your main application to the cloud is securing it, which isn't something you can do while your app is being developed locally. Once in the cloud, connections and permissions between each service are managed via security groups, which are essentially light firewalls around each service instance. Controlling access and security is done with pure configuration, completed with the web console, and is mostly straightforward. Most of the bugs I've encountered with services interacting in AWS have been from misconfigured security groups.
* **AWS Vs. Azure:**
  + Use Azure if you want to run .NET stuff.
  + Use AWS for the other things especially things like lambda functions and APIs Gateway
  + AWS have advantage over azure in the availability and number of data centers around world till now.
  + The cost in nearly same.
* **AWS Vs. Heroku:**
  + Heroku is a deadly simple in deployment so it’s very good for the new developers but it also lake a lot of needed options for advanced enterprise use.
  + So use Heroku if the things is simple.
* **AWS Vs. IBM BlueMix:**
  + IBM have a lot of good services like Watson that you may need to integrate with but you shouldn’t use IBM BlueMix if you don’t need portability and if you don’t want services that covered by CloudFoundary.

**Understanding the core services of AWS:**

* **Intro:**



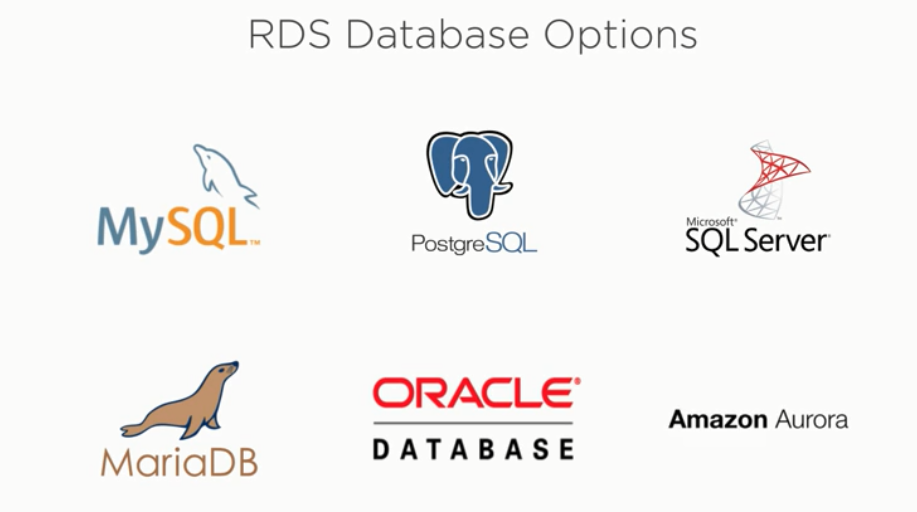
* + EC2 (Elastic Cloud Compute) is the basic unit of computing and you can think in it as a virtual machine where you can run an application.
  + S3 (Simple Storage Service) AWS static file hosting service.
  + RDS (Relational Database Service).
  + Route53 which is the DNS service that empowers your EC2 instances and S3 buckets to be accessible via URLS.
* EC2:
  + Amazon has added many features to EC2 with the time from 2006 till now.
  + What can you do with it?
  + 
  + Elastic cloud compute: Elastic here means that the computing service can be expanded or retracted as needed.
  + The basic building block for EC2 is the EC2 instance, and the instance is essentially a virtual server which operating system agnostic.
  + Demo: First select your image (linux, windows, etc.) from Quick start, marketplace or community AMIs (but take a note that once you installed the image you should take care of security updates and manually apply the updates) then choose the family of your needs which have a type and each type have attributes like CPU and memory etc. like shown below. And then you can configure instances (AWS will create an auto scaling group to allow you easily scaling EC2 images up and down and this is called EC2 for scaling AMIs and it’s great for scaling your self-created images or your third party but to scale application you already installed on an instance this is should instead use Elastic Beanstalk service), storage, tags and security groups configuration to control the IP Mappings from and to the instance and etc.



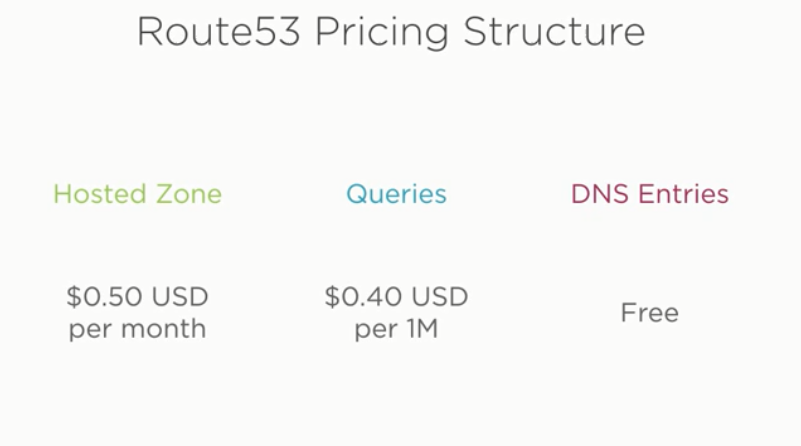
* S3:
* It’s object max size is 5 Tera byte 😊
* Buckets are the foundational structure in S3, it’s your root resource to which you can add, delete, or modify objects. And there are a lot of config options that you can set on buckets like permissions, hosting options and logging. Its also can be configured to trigger events when changes happened, preserve old versions of the object, automatically replicate objects to different regions and much more.
* S3 Buckets are accessed via URLs.
* There are a lot of wonderful things like static website hosting and object level permissions.
  + As we mentioned before, with S3 we can solve the latency problem by replication but there is another simpler and better way by using another amazon service called CloudFront, you can cache your content (Sometimes called edging your content) at a lot of locations around the world.
  + Price Structure:



* RDS:
  + RDS is a collection of AWS services for managed relational databases.
  + We call the databases managed because AWS takes care of all the backups, software updates and etc.
  + You can of course deploy your own database instance on EC2 instance but you will have the responsibility of database backups, redundancy and software patches.
  + Till Now:



* + It’s easy to configure your EC2 instances to run with your RDS databases.
  + It’s dead easy to configure firewall and security and scalability.
  + There is another not relational solutions come out of the RDS like DynmoDB (No-SQL) and RedShift (Data Warehouse).
* Route53:
  + Route53 is Amazon's service for DNS management both inside and outside AWS.
  + It allows you to easily configure domain names to resolve to internal AWS services.
  + You can use domain names that you already own or can register new ones through AWS.
  + DNS stands for Domain Name System, and the purpose is to translate a human readable URL into an IP address so that two computers can connect. EC2 instances can be configured with public IP addresses, but certain Amazon resources like S3 buckets or load balancers are a little more complicated and don't have static visible IP addresses. That's where Route53 comes into the picture, allowing you to set up URL resolution to AWS resources directly, bypassing any need for you to see an IP. Of course, behind the scenes, IP resolution is happening, but Route53 conveniently abstracts this away.
  + Cost Structure:



* + There are a few more features that Route53 provides. One of the most useful that I've found is the health checks service. This allows you to set up regular checks for a given URL path. Health checks will send you alerts based on different rules, for instance, if the URL request gets a response of 503 or 404. This is an essential tool if you have a customer facing web application, and at $0.50 per month for each alarm, it's not hard to justify the value of something like this.

**Enhancing Your App with AWS Databases and Application Services:**

* Intro:



* Elastic Beanstalk is an application service that will take your application written in Java, Python, Node, Ruby, or any other language and get it running on an EC2 instance.
* DynamoDB and RedShift are two managed database solutions offered by Amazon that are a little bit different than those in RDS. DynamoDB is a managed NoSQL document database service, and RedShift is a managed data warehouse solution. Both, due to being managed by Amazon, offer many benefits over rolling your own solution.
* Virtual Private Cloud is a service enabling you to control access and secure your instances and services in AWS. It's used by almost every service in AWS and is the best way to ensure that your data and application is safe.
* CloudFront is a CDN service available from AWS, and it enables you to edge your files so users can get to them superfast. It can be used in conjunction with S3 as mentioned previously, or with a web application behind a load balancer.
* CloudWatch is a logging and alarm service used with almost every AWS service available. You can set up alarms to let you know when instances are down, EC2's CPU is high, or when S3 charges reach a threshold, for some examples. There's any number of things you can do with CloudWatch.