AWS for PL/SQL developers

All AWS infrastructure can be provisioned via the AWS console, via APIs, and via code (IAC). The basic building block is the EC2 instance (EC2 standing for Elastic Compute Cloud). You can think of this as essentially being synonymous with a VM - you pick out the size you want in terms of compute (IOPs etc) and a certain amount of direct-attached storage, and a few seconds later it's available for you to start playing with. The fact that this can be done is code or via an API means that we can programmatically spin up a new EC2 instance ('VM') whenever we want, have it available to us for however long we want it for, and then we destroy it ('tear it down').

A short summary of some of the other AWS technologies that are commonly used at RI:

AWS Service	What it does	Documentation Links
Amazon Aurora	Amazon Aurora is a MySQL and PostgreSQL compatible relational database built for the cloud, that combines the performance and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases.	• Info >> • Documentation >>
Amazon DynamoDB	Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models.	• Info >> • Documentation >>
AWS Certificate Manager	AWS Certificate Manager is a service that lets you easily provision, manage, and deploy Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services.	• Info >> • Documentation >>
AWS Direct Connect	AWS Direct Connect enables a dedicated private network connection from your premises to AWS.	• Info >> • Documentation >>
Amazon CloudFront	Amazon CloudFront is a global content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to your viewers with low latency and high transfer speeds.	• Info >> • Documentation >>
Amazon CloudWatch	Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources.	• Info >> • Documentation >>
Amazon Route 53	Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service.	• Info >> • Documentation >>
Amazon EC2	Provides the virtual application servers, known as instances, to deploy your code on.	• User Guide for Linux Instances >> • User Guide for Windows Instances >> • API Reference >>
Amazon EC2 Container Service	A highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.	• User Guide >> • API Reference >> • CLI Reference >>
Elastic Load Balancing	Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses.	• Info >> • Documentation >>
AWS Identity and Access Management (IAM)	AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.	• Info >> • Documentation >> • Create IAM account >>

Amazon Kinesis	Amazon Kinesis is a platform for streaming data on AWS, making it easy to load and analyse streaming data, and also providing the ability for you to build custom streaming data applications for specialised needs.	• Info >> • Documentation >>
AWS Lambda	Runs code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration.	• Developer Guide >>
AWS CloudFormation	An easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion.	• User Guide >> • API Reference >> • CLI Reference >>
Amazon S3	Amazon Simple Storage Service is an object storage built to store and retrieve any amount of data from anywhere.	• Intro >> • Documentation >>
AWS Config	Provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance.	• User Guide >> • API Reference >> • CLI Reference >>
Auto Scaling	Auto Scaling helps you maintain application availability and allows you to dynamically scale your Amazon EC2 capacity up or down automatically according to conditions you define.	• Intro >> • Documentation >>
Amazon Simple Email Service (SES)	Amazon Simple Email Service (Amazon SES) is a cloud-based email sending service designed to help digital marketers and application developers send marketing, notification, and transactional emails.	• Info >> • Documentation >>
Amazon Simple Notification Service (SNS)	Amazon Simple Notification Service (SNS) is a flexible, fully managed pub/sub messaging and mobile notifications service for coordinating the delivery of messages to subscribing endpoints and clients.	• Info >> • Documentation >>
Amazon Simple Queue Service (SQS)	Amazon Simple Queue Service (SQS) is a fully managed message queuing service that makes it easy to decouple and scale microservices, distributed systems, and serverless applications.	• Info >> • Documentation >>
Amazon Virtual Private Cloud (VPC)	Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (AWS) cloud where you can launch AWS resources in a virtual network that you define.	• Info >> • Documentation >>

10 minute tutorials:

https://aws.amazon.com/getting-started/tutorials/

CloudAcademy AWS Learning Path:

https://cloudacademy.com/learning-paths/platforms/amazon-web-services/

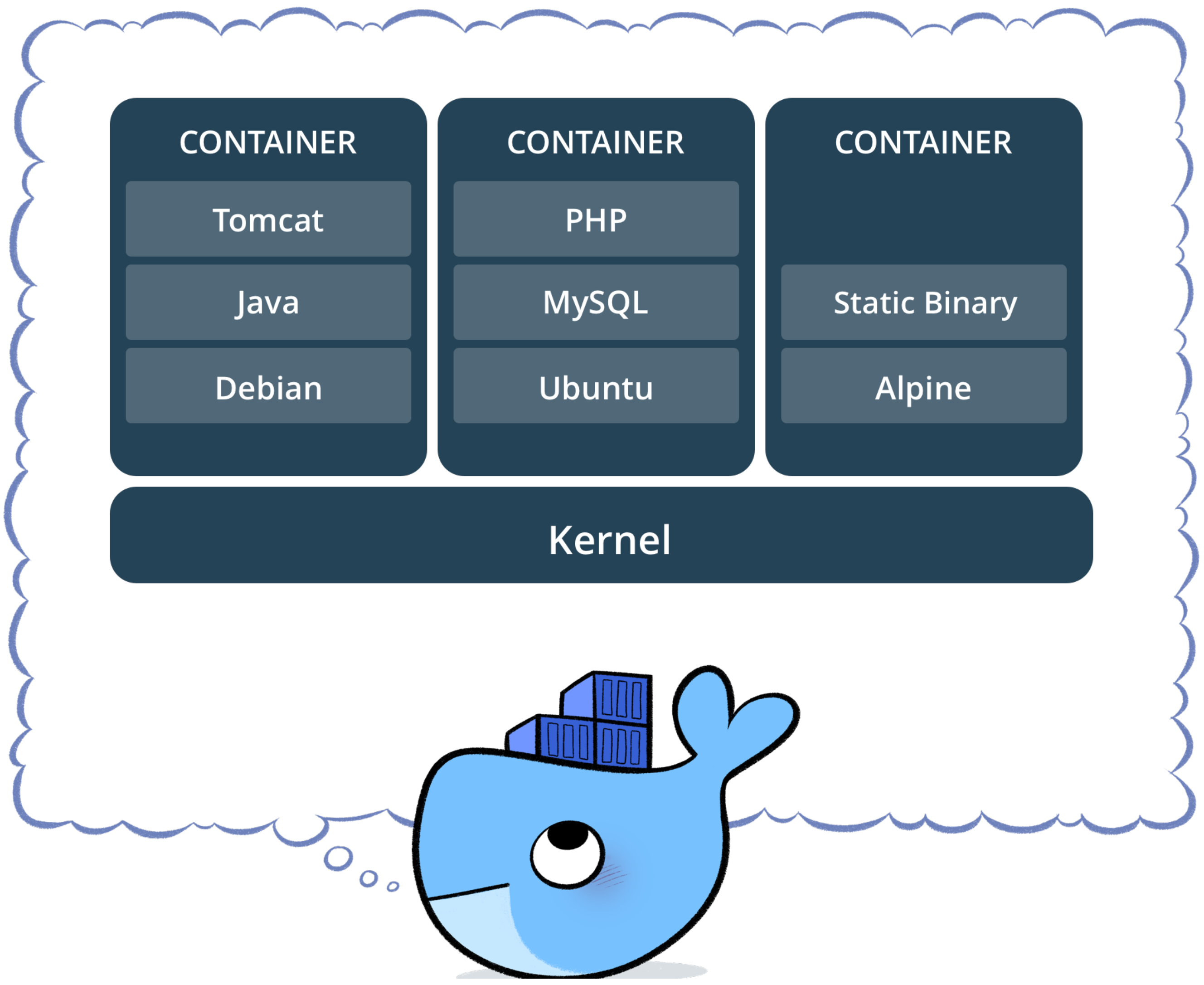
$59 per person per month (possible reduction if buying multiple).

Amazon DBs

See Which AWS DB to use and when

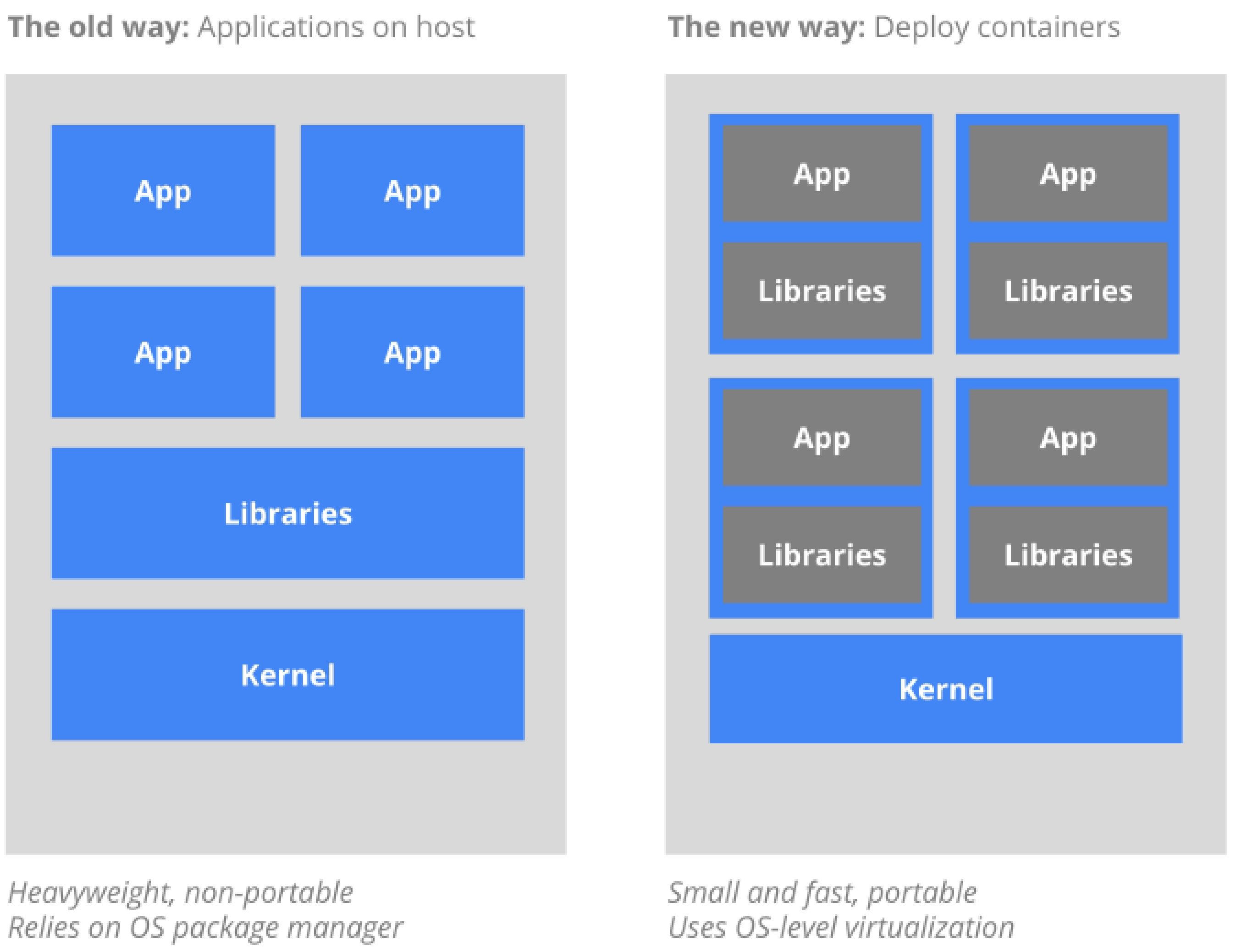
Containers and Docker

Docker is a type of containerisation application - in RI you can treat these two terms as being synonymous. A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries and settings. Containers isolate software from its surroundings, for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure.



The *Old Way* to deploy applications was to install the applications on a host using the operating system package manager. This had the disadvantage of entangling the applications' executables, configuration, libraries, and lifecycles with each other and with the host OS. One could build immutable virtual-machine images in order to achieve predictable rollouts and rollbacks, but VMs are heavyweight and non-portable.

The *New Way* is to deploy containers based on operating-system-level virtualization rather than hardware virtualization. These containers are isolated from each other and from the host: they have their own filesystems, they can't see each others' processes, and their computational resource usage can be bounded. They are easier to build than VMs, and because they are decoupled from the underlying infrastructure and from the host filesystem, they are portable across clouds and OS distributions.


The old way: Applications on host
App
App
App
App
Libraries
Kernel
Heavyweight, non-portable
Relies on OS package manager
The new way: Deploy containers
App
Libraries
App
Libraries
App
Libraries
App
Libraries
Kernel
Small and fast, portable
Uses OS-level virtualization

Because containers are small and fast, one application can be packed in each container image. This one-to-one application-to-image relationship unlocks the full benefits of containers. With containers, immutable container images can be created at build/release time rather than deployment time, since each application doesn't need to be composed with the rest of the application stack, nor married to the production infrastructure environment. Generating container images at build/release time enables a consistent environment to be carried from development into production. Similarly, containers are vastly more transparent than VMs, which facilitates monitoring and management. This is especially true when the containers' process lifecycles are managed by the infrastructure rather than hidden by a process supervisor inside the container. Finally, with a single application per container, managing the containers becomes tantamount to managing deployment of the application.

Summary of container benefits:

- **Agile application creation and deployment**: Increased ease and efficiency of container image creation compared to VM image use.
- **Continuous development, integration, and deployment**: Provides for reliable and frequent container image build and deployment with quick and easy rollbacks (due to image immutability).
- **Dev and Ops separation of concerns**: Create application container images at build/release time rather than deployment time, thereby decoupling applications from infrastructure.
- **Environmental consistency across development, testing, and production**: Runs the same on a laptop as it does in the cloud.
- **Cloud and OS distribution portability**: Runs on Ubuntu, RHEL, CoreOS, on-prem, Google Container Engine, and anywhere else.
- **Application-centric management**: Raises the level of abstraction from running an OS on virtual hardware to run an application on an OS using logical resources.
- **Loosely coupled, distributed, elastic, liberated micro-services**: Applications are broken into smaller, independent pieces and can be deployed and managed dynamically – not a fat monolithic stack running on one big single-purpose machine.
- **Resource isolation**: Predictable application performance.
- **Resource utilization**: High efficiency and density.