ORACLE
Cloud Infrastructure

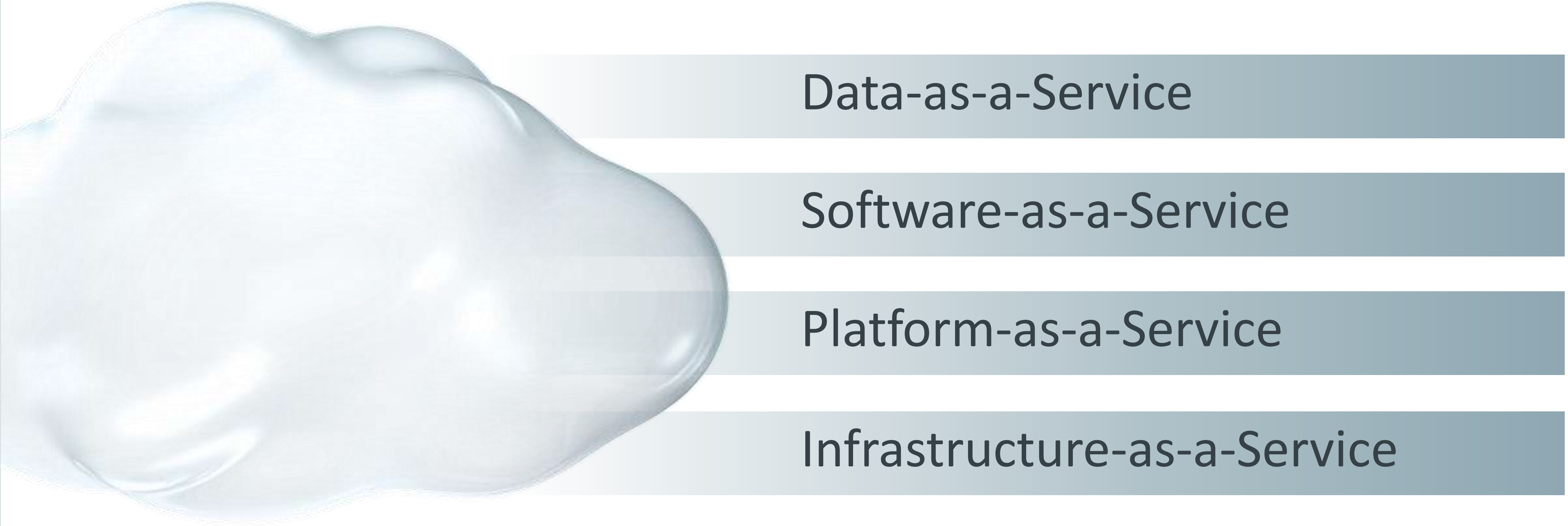
Oracle Cloud Infrastructure Overview

Sárecz Lajos

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Cloud



Data-as-a-Service

Software-as-a-Service

Platform-as-a-Service

Infrastructure-as-a-Service

Oracle Infrastructure as a Service: Mission

Enable customers

- To run any type of workload in the cloud
- To run Oracle workloads in the most optimized way

ORACLE

Oracle IaaS: Competitive Differentiators

Predictable Performance

Ease of Migration

Control and Visibility

Choice of Deployment

Enterprise Workloads, Service & Support

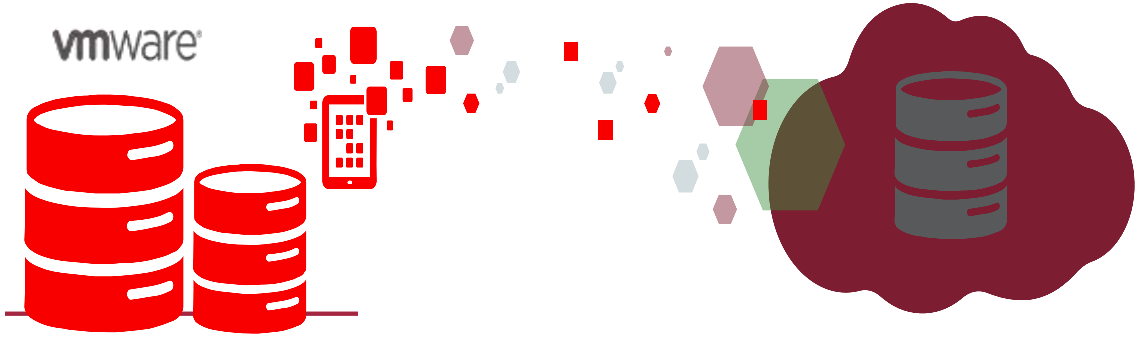
Predictable Performance

Run enterprise workloads with consistent performance

- Next-Gen Data Center Technology
- Single-Tenant Deployment Options
- No Oversubscription
- Benefits entire stack

Ease of Migration

“Application Capsule” approach enables true Virtualized to Cloud (V2C) lift-n-shift



1 Apps Lift & shift

- No complex, manual migration.
- Application stays the same (VMware VMs, networking, etc.)

2 Save 40-60%

- Relative to running in a virtualized data center
- No cost and effort of re-work

3 Increase Agility

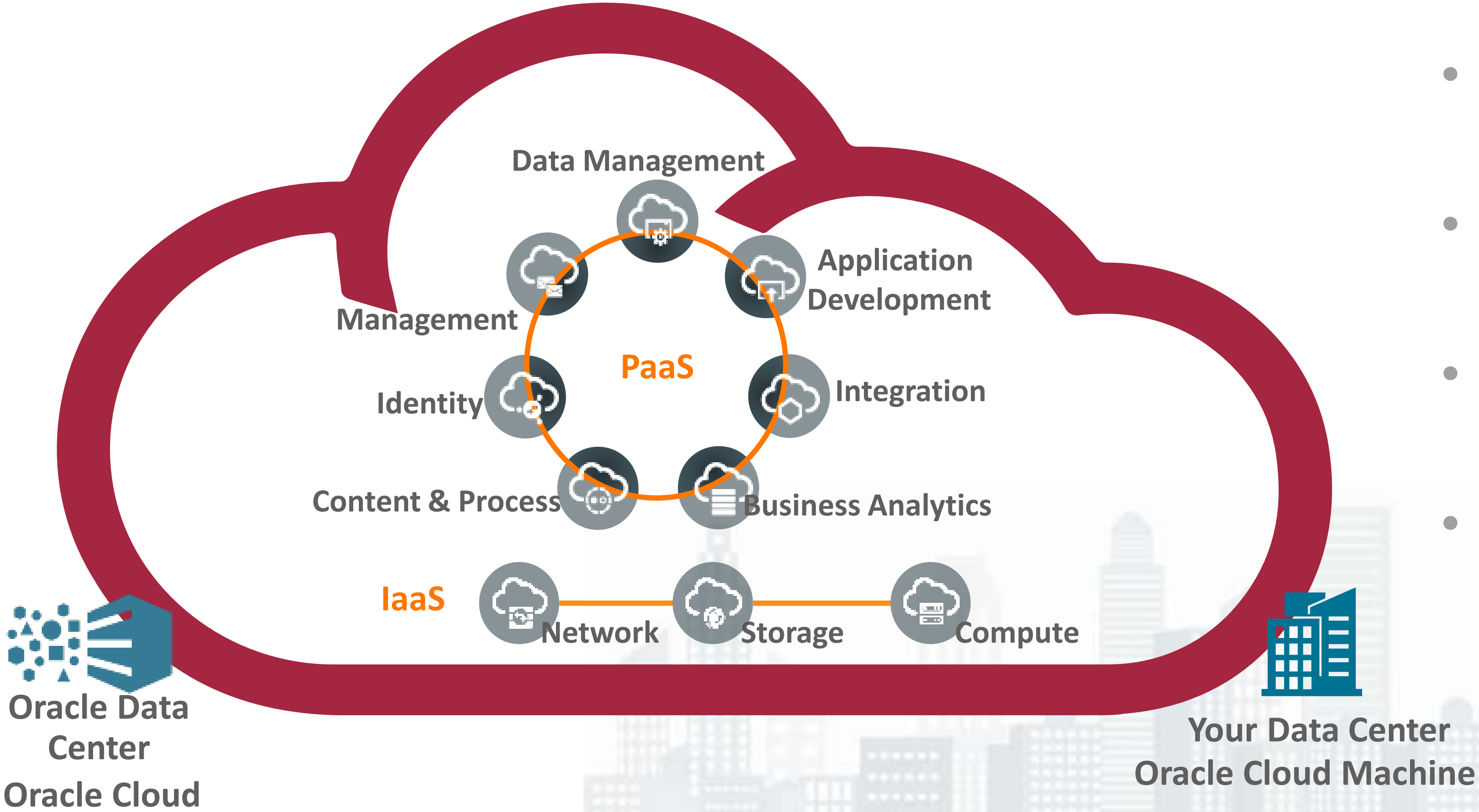
- Spin up entire environments (‘00s of VMs) in ~5 mins
- Scale in any geographic region

ORACLE | ravello

ORACLE

Choice of Deployment

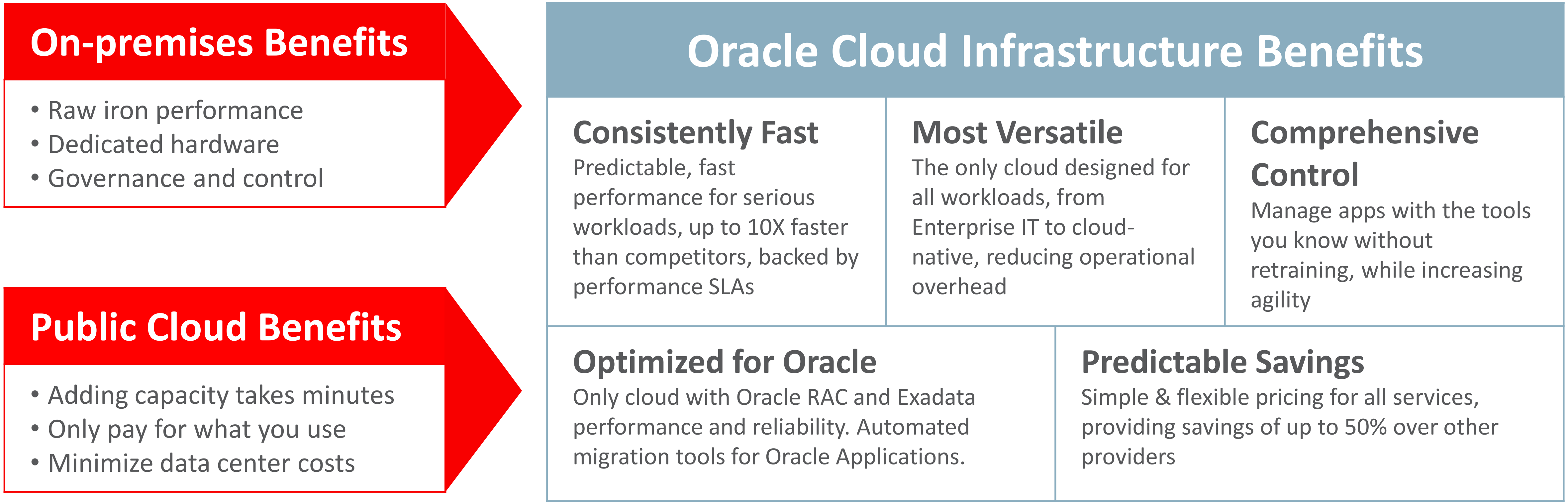
Cloud@Customer: *Oracle Cloud Services behind your firewall*



- Same PaaS and IaaS software
- Same updates as Oracle Cloud
- Same subscription and pay-as-you-go pricing
- Single vendor for the entire solution

ORACLE

Oracle Cloud Infrastructure Benefits: Best of Both Worlds



On-premises Benefits

- Raw iron performance
- Dedicated hardware
- Governance and control

Public Cloud Benefits

- Adding capacity takes minutes
- Only pay for what you use
- Minimize data center costs

Oracle Cloud Infrastructure Benefits

Consistently Fast

Predictable, fast performance for serious workloads, up to 10X faster than competitors, backed by performance SLAs

Most Versatile

The only cloud designed for all workloads, from Enterprise IT to cloud-native, reducing operational overhead

Comprehensive Control

Manage apps with the tools you know without retraining, while increasing agility

Optimized for Oracle

Only cloud with Oracle RAC and Exadata performance and reliability. Automated migration tools for Oracle Applications.

Predictable Savings

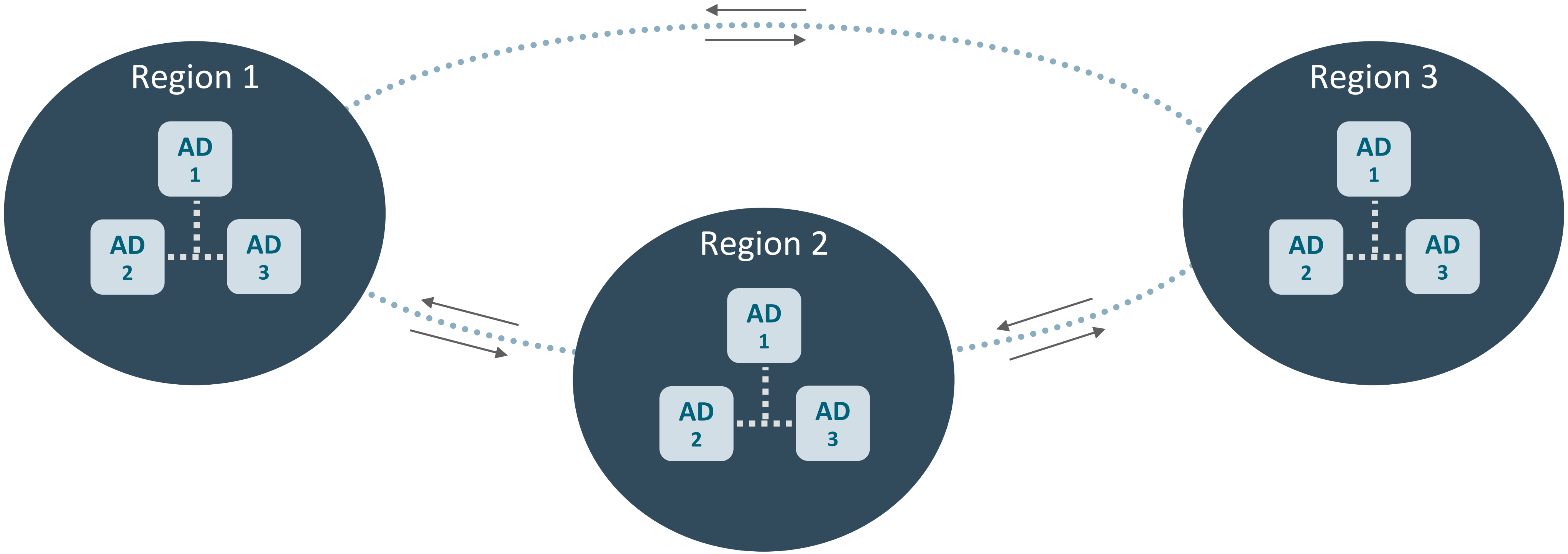
Simple & flexible pricing for all services, providing savings of up to 50% over other providers

Latest Technologies Enable a Modern Cloud Infrastructure

Technology	Benefit
Availability domains	Enables enterprise-level high availability
Flat, non-blocking network	Enables predictable low latency; eliminates “noisy neighbors”
Off-box IO virtualization & automated hardware wiping	Enables secure deployments of bare metal servers without Oracle management software overhead
Direct-attached NVMe storage	Enables highest IO workloads

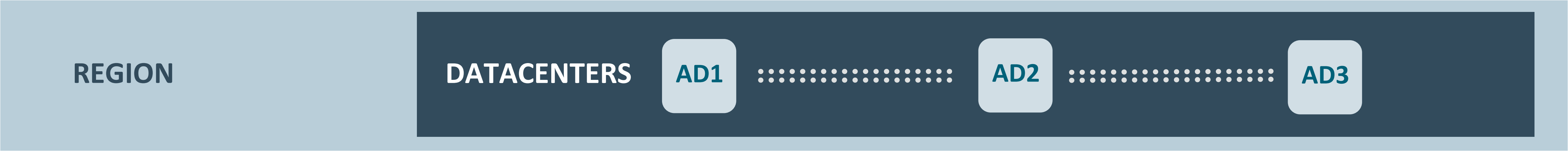
Region / Availability Domain Topology

- Regions serve different geographies, provide disaster recovery
- Availability Domains (ADs) provide a high availability foundation in a Region



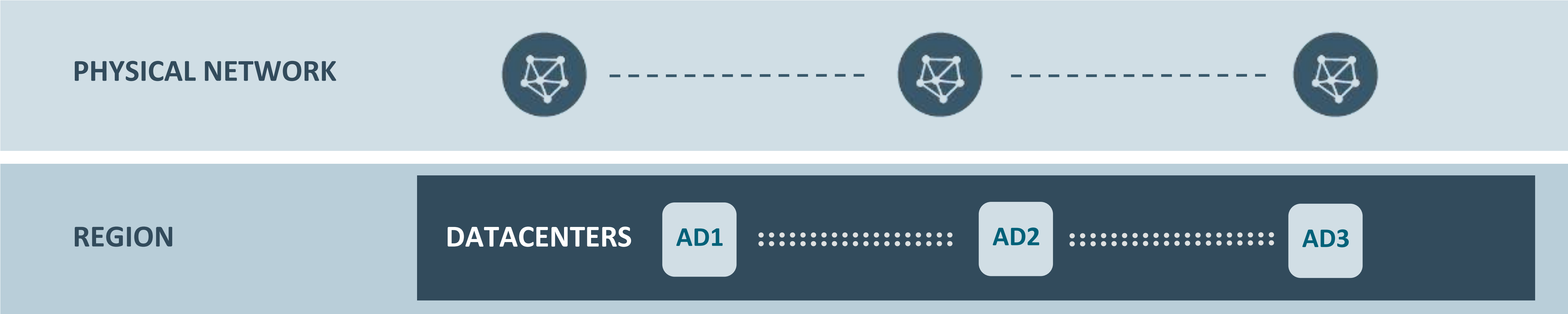
Inside a Region – High Availability Building Blocks

- Multiple fault-decorrelated, completely independent datacenters: ADs
- Predictable low latency & high speed, encrypted interconnect between ADs
  - < 500µs expected one-way latency, 1Tb/s bandwidth
- Enables zero-data-loss architectures (e.g. Oracle MAA) and high availability scale-out architectures (e.g. Cassandra)



Inside an AD – High Scale, High Performance Network

- Non-oversubscribed network – flat, fast, predictable
- Very high scale – ~1 million network ports in an AD
- Predictable low latency & high speed interconnect between hosts in an AD
  - < 100µs expected one-way latency within an AD, 25Gb/s bandwidth

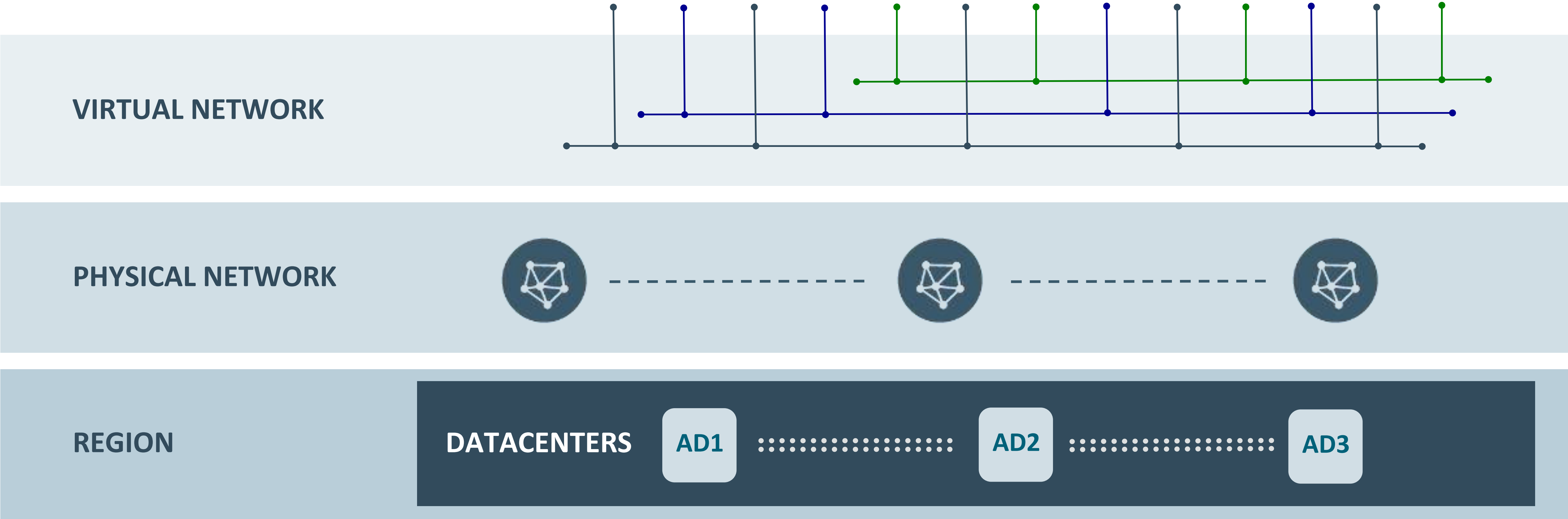

PHYSICAL NETWORK

REGION

DATACENTERS

Comprehensive Virtual Network with Off-box Virtualization

- Highly configurable private overlay networks – moves management and IO out of the hypervisor and enables lower overhead and bare metal instances



VIRTUAL NETWORK

PHYSICAL NETWORK

REGION

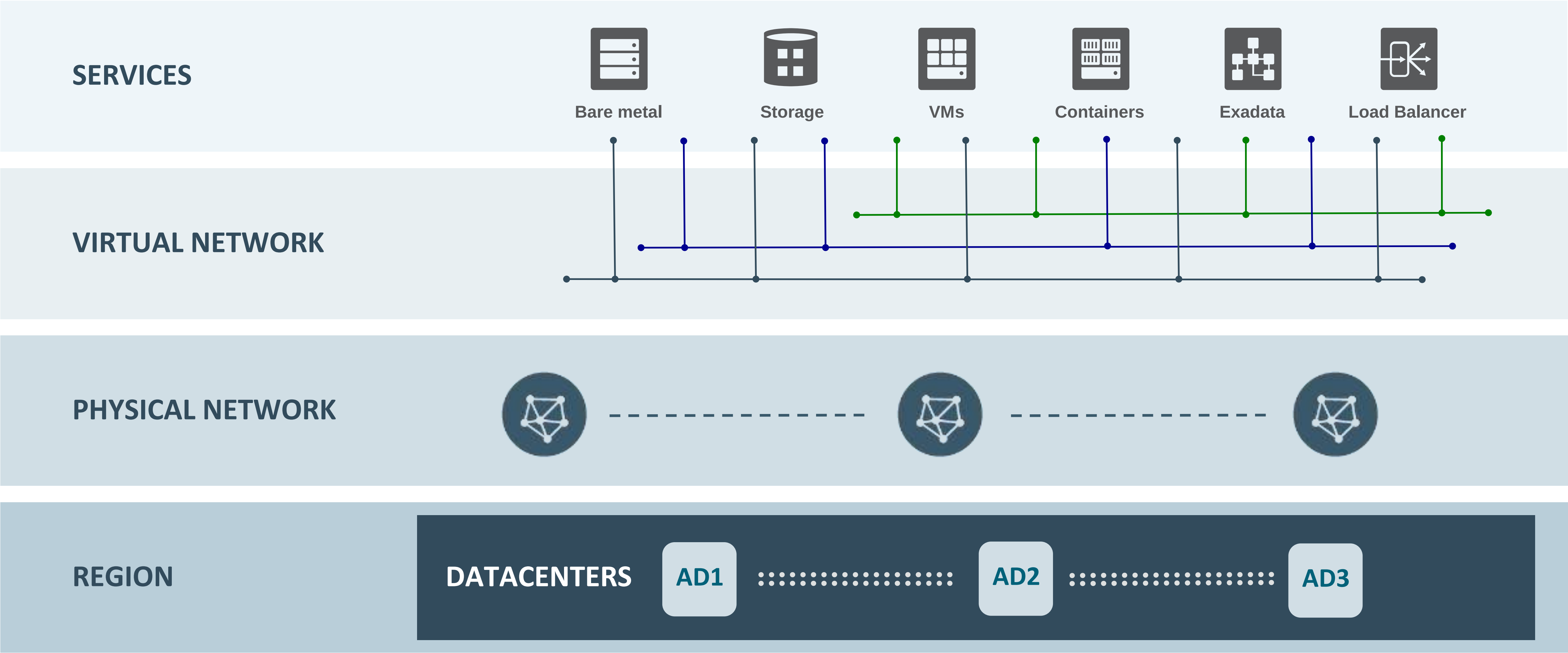
DATACENTERS

AD1

AD2

AD3

Technology-Enabled Cloud Infrastructure Innovation



SERVICES

Bare metal

Storage

VMs

Containers

Exadata

Load Balancer

VIRTUAL NETWORK

PHYSICAL NETWORK

REGION

DATACENTERS

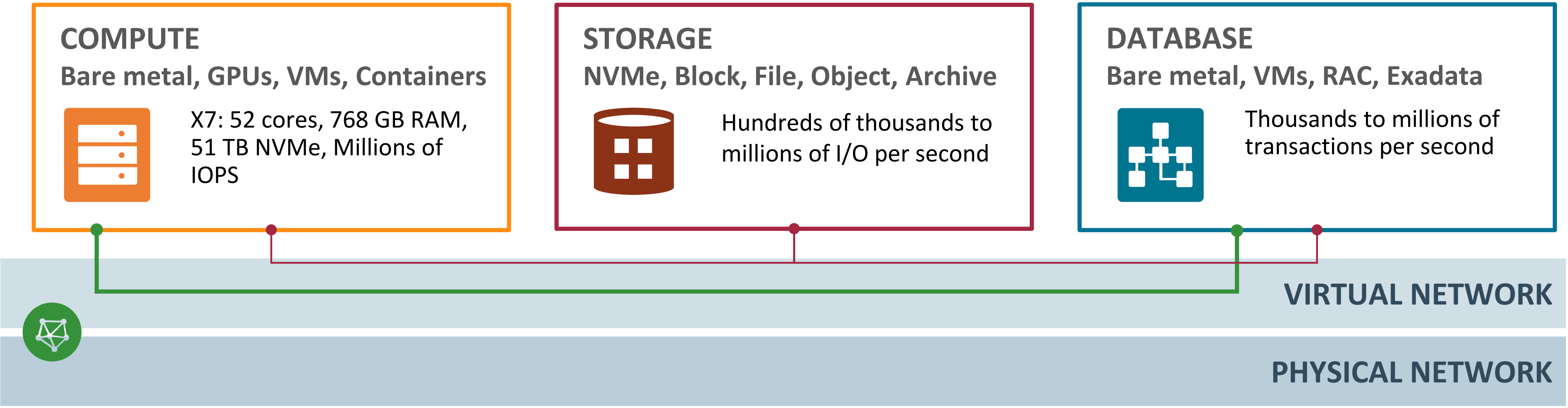
AD1

AD2

AD3

High Performance Cloud Services: Compute, Storage, Database

- Low latency, high bandwidth networks and up to 97% lower-cost connectivity
- Highly configurable virtual networking, load balancing, firewalls, DNS
- Superfast and predictable compute, database, and up to 98% lower cost storage



COMPUTE

Bare metal, GPUs, VMs, Containers

X7: 52 cores, 768 GB RAM, 51 TB NVMe, Millions of IOPS

STORAGE

NVMe, Block, File, Object, Archive

Hundreds of thousands to millions of I/O per second

DATABASE

Bare metal, VMs, RAC, Exadata

Thousands to millions of transactions per second

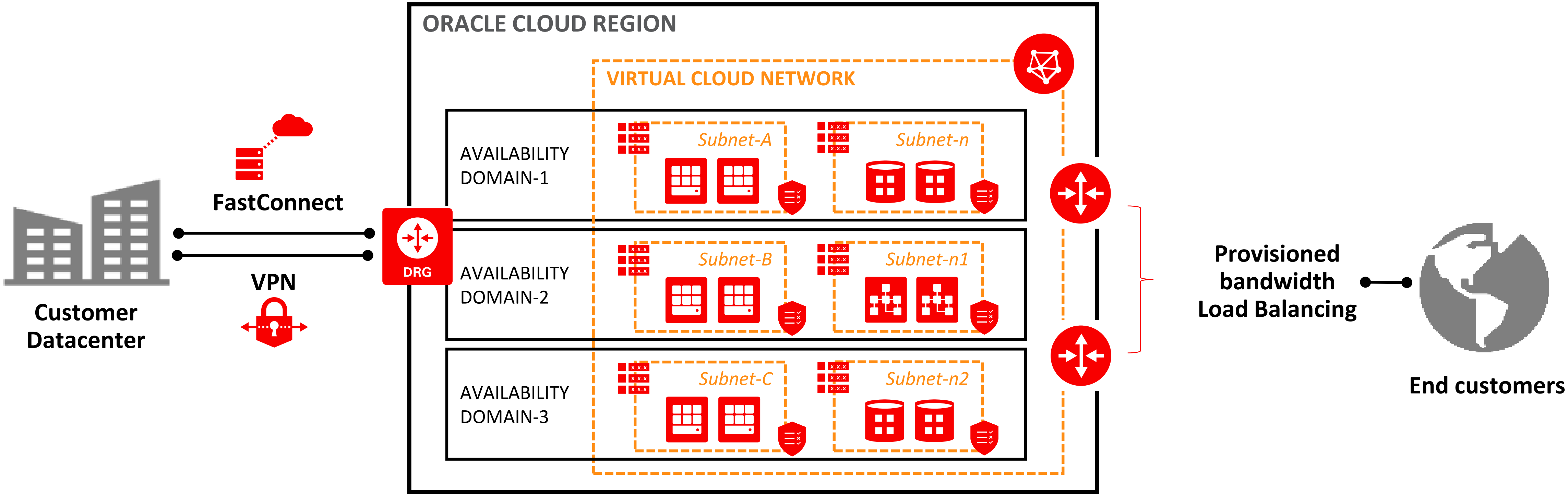
VIRTUAL NETWORK

PHYSICAL NETWORK

Virtual Network: High-Fidelity Private Networking and Connectivity

Secure, reliable connectivity: IPSec VPN, FastConnect

Deep VCN control: Subnets, routing rules, IP address space, firewall rules



Console or API-driven; same 25Gbps network for all core services; <500µs one-way latency between Availability Domains

Oracle Cloud Infrastructure and Kubernetes

Roll Your Own, Pre-Built Installer, Managed Service

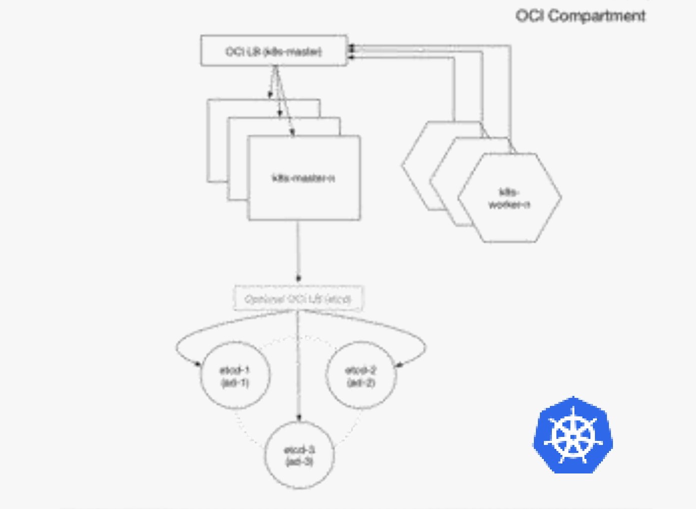
OCI

DIY Container Management

IaaS

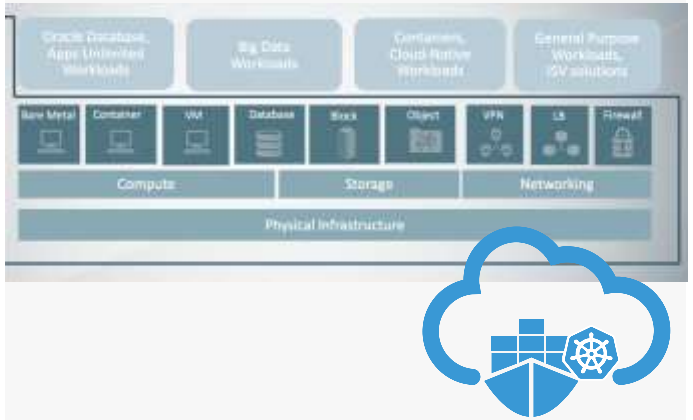
Quickstart Experience

OSS Terraform Installer on GitHub



Self Managed Kubernetes Service

OCI Container Engine for Kubernetes



Enterprise Class Managed Kubernetes Service

CaaS

Constraints that Prevent Consumption of Public Cloud

DATA SOVEREIGNTY

- Comply with regulatory, legal and privacy requirements
- Sensitive data on premises
- Custom security standards

CONTROL

- Keep control over business-critical systems
- Use your own firewalls, load balancers, hardware VPNs, etc.
- Extremely high SLAs

LATENCY

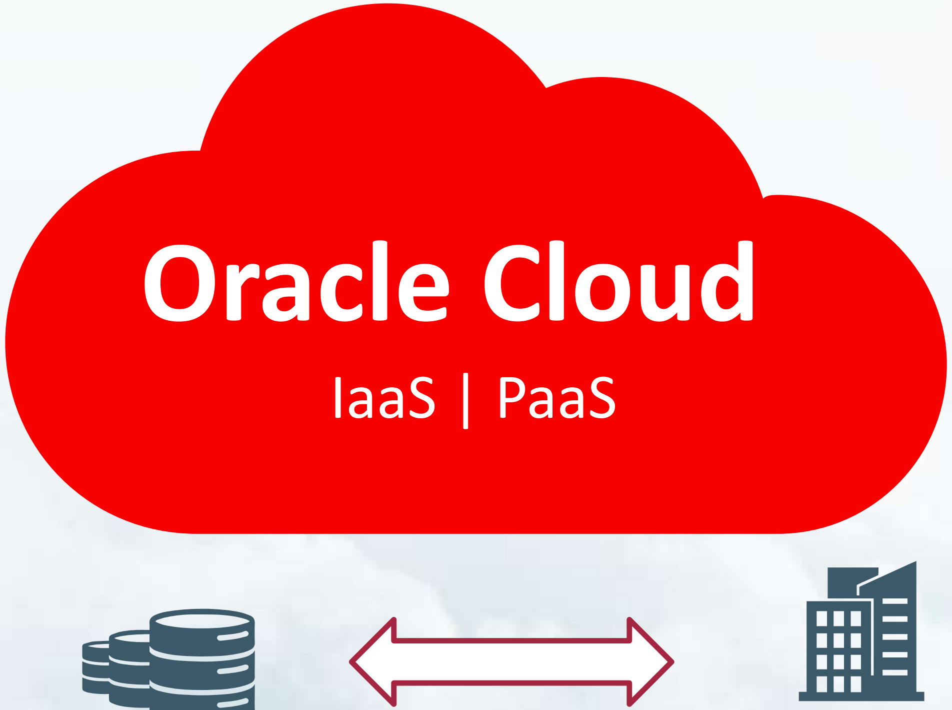
- Connect with back-end mainframes, databases, ERPs, etc. with near zero latency
- Dedicated infrastructure offers lower latency

ORACLE

Cloud At Customer | Oracle Cloud Behind Your Firewall

Complete deployment choice

- Same Oracle Cloud
- Simple subscription (IaaS, PaaS) and Pay-as-You-Go (PaaS) pricing
- Oracle owns hardware and software; manages the platform remotely
- Leave the stack to Oracle. Focus on using the services
- Single vendor handshake



Oracle Cloud

IaaS | PaaS

Oracle Data Center
Oracle Cloud

Your Data Center
Oracle Cloud at Customer

- Oracle Cloud At Customer
- Exadata Cloud At Customer
- Big Data Cloud At Customer

Cloud At Customer | Building Blocks

Common Control Plane

Control plane

- All Cloud Services managed through shared Control Plane to optimize cost and scale
- Ability to flexibly add compute and storage capacity as needed in granular increments

Oracle Cloud At Customer

Building Blocks

Compute: X6-2 server
4xHDD, **4xNVMe**
40 cores

Block Storage: ZFS (45 TB)

Object Storage: Object Storage (128TB)

IaaS

+

Oracle PaaS

Exadata Cloud At Customer

Configurations

Full Rack

Half Rack

Quarter Rack

Eighth Rack

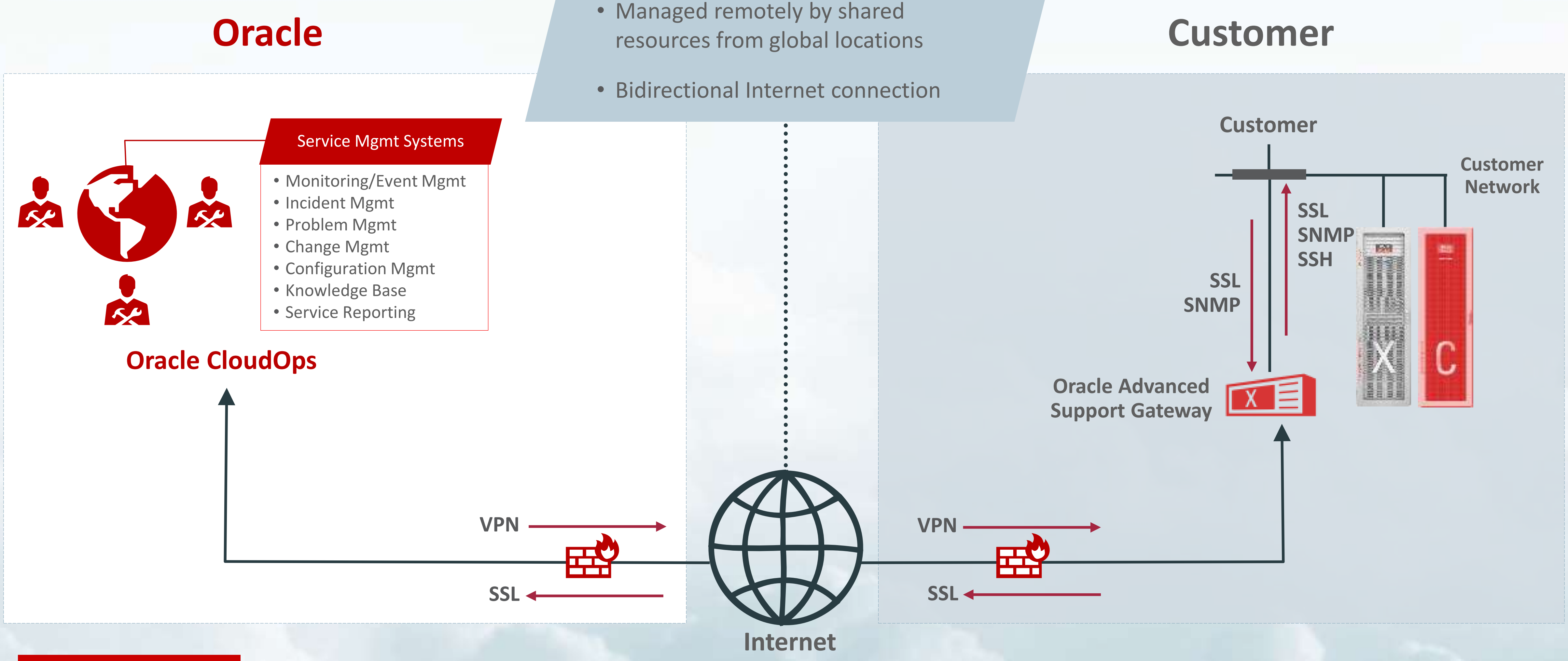
Cloud at Customer is Not a Private Cloud

Cloud at Customer		Private Cloud
✓	Subscription-Based, Consumption-based Pricing	NO
✓	Public Cloud Services Behind Your Firewall	NO
✓	Compatibility with public Oracle Cloud	NO
✓	Fully Managed Cloud by Oracle	NO

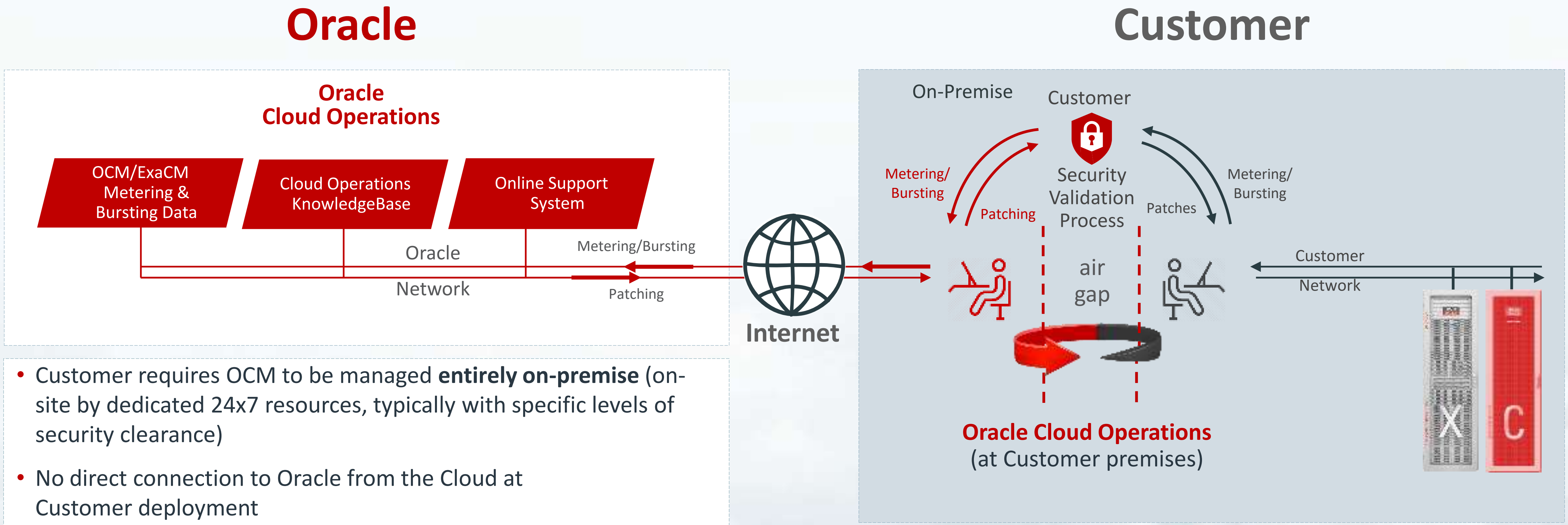
ORACLE

Standard (Connected) Cloud Operations

- Managed remotely by shared resources from global locations
- Bidirectional Internet connection

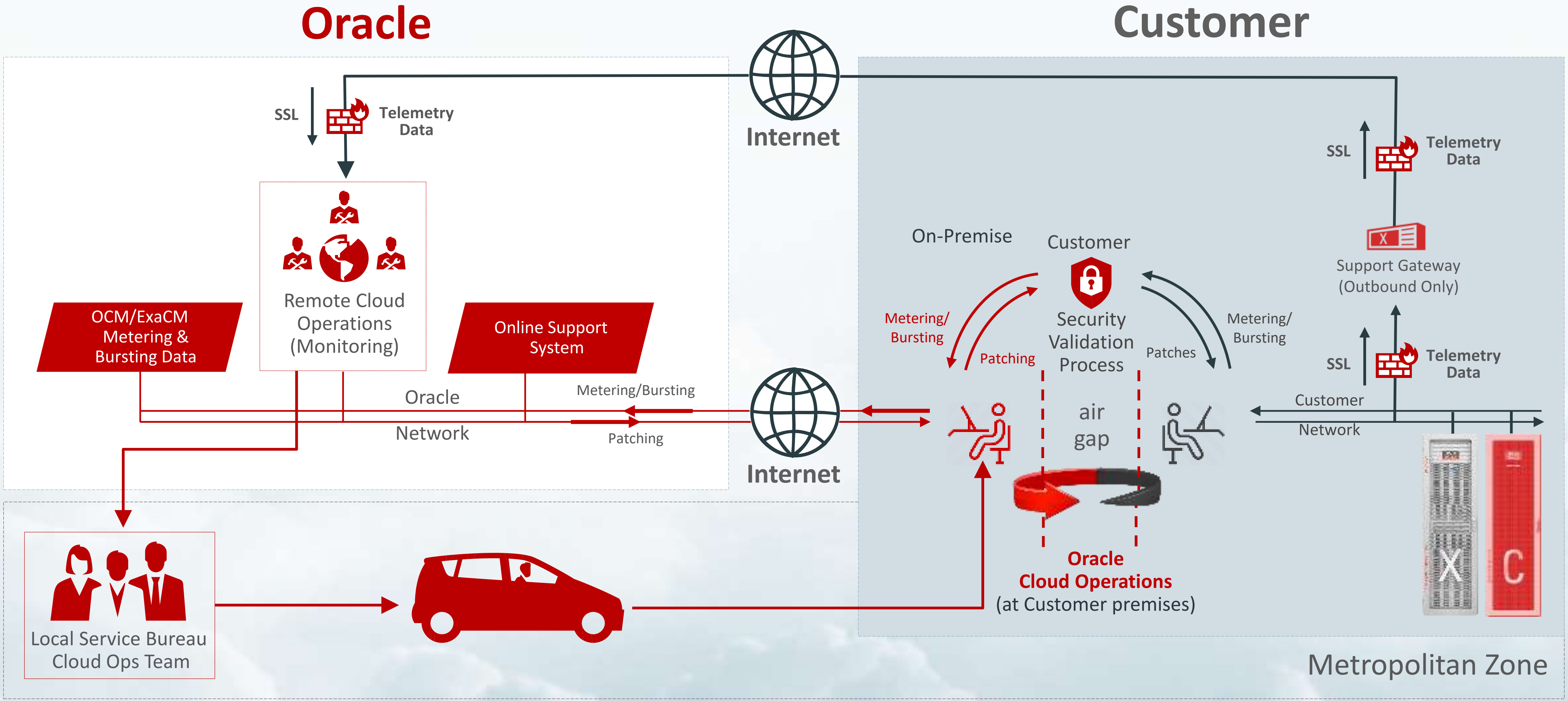


Oracle Cloud at Customer: Disconnected Mode



- Customer requires OCM to be managed **entirely on-premise** (on-site by dedicated 24x7 resources, typically with specific levels of security clearance)
- No direct connection to Oracle from the Cloud at Customer deployment
- Controlled ability to move metering/troubleshooting data on/off site
- “Swivel Chair” approach used by dedicated on-site Cloud Operations team

Oracle Cloud at Customer: Semi-Connected Mode



Cloud Orchestration & Infrastructure as Code (IaC)

- Infrastructure Lifecycle
  - Provision
  - Update
  - Destroy
- The 4 Broad categories of IAC:
  - Ad hoc scripts
  - Configuration management tools (chef, puppet, …)
  - Server templating tools (Packer, Vagrant, Docker, …)
  - Server provisioning tools (Terraform, cloud formation, heat)

Terraform – 0.11.7 – built by HashiCorp

- Written in Go
- Runtimes available for OSX, FreeBSD, Linux, OpenBSD, Solaris, Windows
- Mac OS X, FreeBSD, Linux, OpenBSD, Solaris, Windows
- Fast development – releases monthly+
- HCL Hashi Configuration Language
  - – JSON interoperable
- HCL - simple markup format
- Plays nice with existing tools - puppet, chef, ansible, etc

Comparison of Terraform to Ansible and CloudFormation

	Terraform	CloudFormation	Ansible
Syntax	HCL	JSON	YAML
Manage Existing Deployments	Difficult	No	Yes
State Management	Yes	No	Yes
Third Party Providers	65+	No	Many++
Infrastructure	Immutable	Immutable	Mutable
Agent/Master	No / No	No / No	No / No
Type	Declarative	Declarative	Procedural

Getting Started with Terraform

- Download
  - binary, apt, yum, choco, brew
- Create a .tf file in a workspace
- hw.tf
- output "hw" {
- value = "test” }
- $ terraform apply
- Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
- Outputs:
- hw = test
- Providers... ->

```
./
├── terraform
├── terraform-provider-atlas
├── terraform-provider-aws
├── terraform-provider-azure
├── terraform-provider-azurerm
├── terraform-provider-chef
├── terraform-provider-cloudflare
├── terraform-provider-cloudstack
├── terraform-provider-consul
├── terraform-provider-opc
├── terraform-provider-oci
```

alicloud archive arukas atlas aws azure azurerm bitbucket chef circonus clc cloudflare cloudstack cobbler consul datadog digitalocean dme dns dnsimple docker dyn external fastly github gitlab google grafana heroku http icinga2 ignition influxdb kubernetes librato local logentries mailgun mysql newrelic nomad ns1 oneandone opc openstack opsgenie packet pagerduty postgresql powerdns profitbricks rabbitmq rancher random rundeck scaleway softlayer spotinst statuscake template terraform tls triton ultradns vault vcd vsphere

HCL – Basic Terraform .tf Format

- Terraform configuration is written into files named .tf files.
- It is based on the HashiCorp Configuration Language (HCL) https://github.com/hashicorp/hcl
- JSON is supported for code generation purposes.
- Most of the configuration takes the form:

```
keyword1 "some_name" {
  key = "value"
  nested {
     key = "value'
   }
}
```

Terraform – Providers

- First thing to do is to use a provider
- Providers abstract the APIs from any given third party in order to create infrastructure.
- Example:

```
provider "oci" {
  tenancy_ocid     = "${var.tenancy_ocid}"
  user_ocid        = "${var.user_ocid}"
  fingerprint      = "${var.fingerprint}"
  private_key_path = "${var.private_key_path}"
  region           = "${var.region}"
}
```

- The baremetal provider enables Terraform to create, manage and destroy resources in your tenancy on OCI.

Terraform – Resources

- Once a provider is configured we can start using that providers resources.
- With the OCI provider, we can start creating instances, block and object storage, networks, etc.
- The following example starts an instance:

```
resource "oci_core_instance" "TFInstance" {
  availability_domain =
"${lookup(data.oci_identity_availability_domains.ADs.availability_domains[var.AD - 1],"name")}"
  compartment_id      = "${var.compartment_ocid}"
  display_name        = "TFInstance"
  hostname_label      = "instance1"
  image               = "${lookup(data.oci_core_images.OLImageOCID.images[0], "id")}"
  shape               = "${var.InstanceShape}"
  subnet_id           = "${var.SubnetOCID}"
  metadata {
    ssh_authorized_keys = "${var.ssh_public_key}"
    user_data           = "${base64encode(file(var.BootStrapFile))}"
  }
}
```

ORACLE

Terraform – Planning Phase

- Once we have put together a configuration to try we can dry-run test this with the planning phase.
- "terraform plan" will take the configuration and give a detailed report on which resources will be created, deleted or modified plus identify what dependent resources are effected by these changes.

```
terraform plan -out=plan1
```

- Saving the plan is useful to ensure that all the steps in the plan were actually applied.

Terraform – Apply

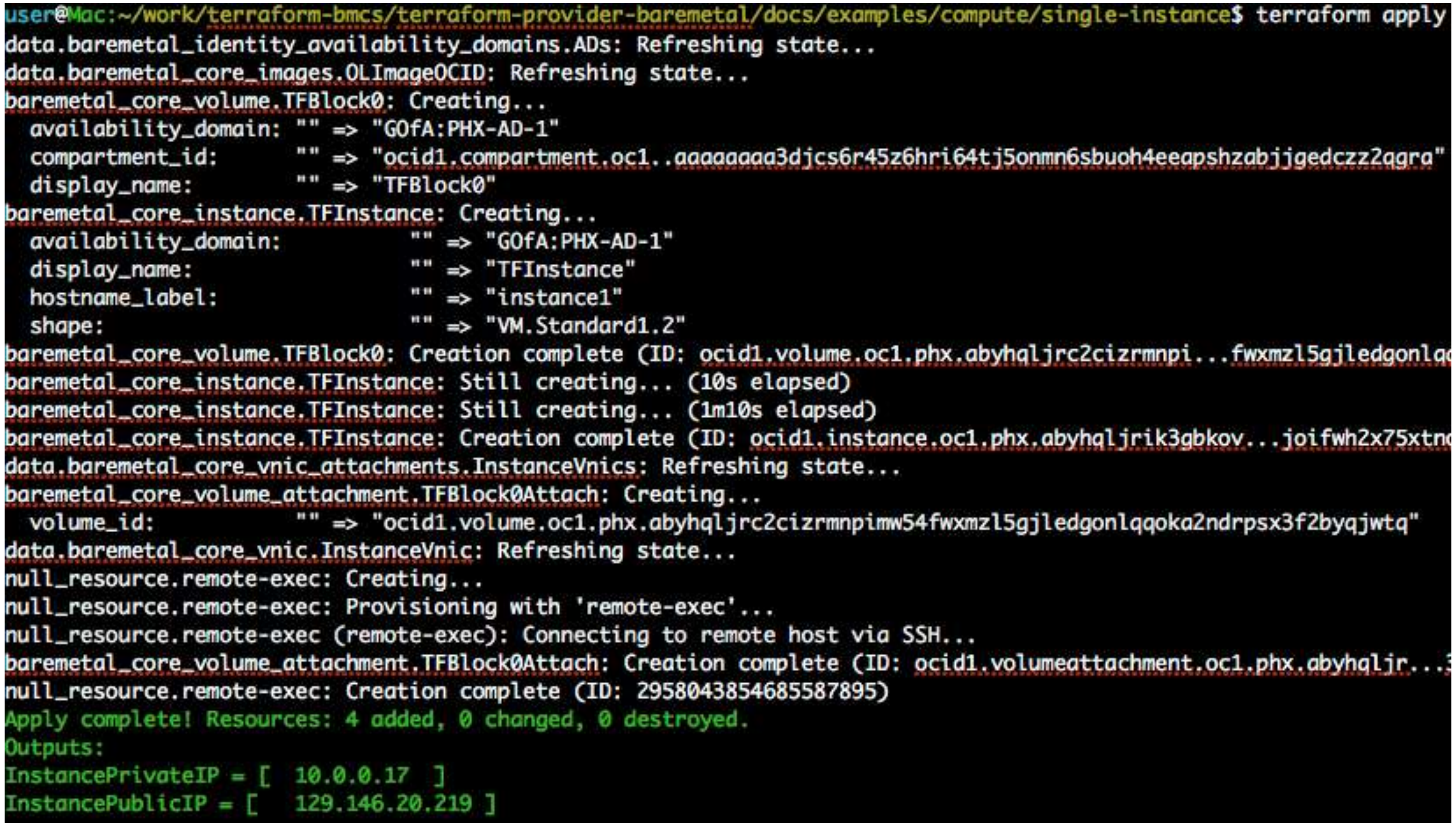
- Once the plan looks good we can go and apply the configuration.

```
$ terraform apply
```

- There is also an option to use saved plans for an apply operation.

```
$ terraform apply plan1
```

- Plan and apply can also target particular resource(s) using the -target flag.
- Plans that are too old will be detected, they are created against a given version of the **terraform.tfstate** file.



```
user@Mac:~/work/terraform-bmcs/terraform-provider-baremetal/docs/examples/compute/single-instance$ terraform apply
data.baremetal_identity_availability_domains.ADs: Refreshing state...
data.baremetal_core_images.OLImageOCID: Refreshing state...
baremetal_core_volume.TFBlock0: Creating...
  availability_domain: "" => "GOfA:PHX-AD-1"
  compartment_id:      "" => "ocid1.compartment.oc1..aaaaaaaa3djcs6r45z6hri64tj5onmn6sbuoh4eeapshzabjjgedczz2qgra"
  display_name:        "" => "TFBlock0"
baremetal_core_instance.TFInstance: Creating...
  availability_domain:          "" => "GOfA:PHX-AD-1"
  display_name:                 "" => "TFInstance"
  hostname_label:               "" => "instance1"
  shape:                        "" => "VM.Standard1.2"
baremetal_core_volume.TFBlock0: Creation complete (ID: ocid1.volume.oc1.phx.abyhqljrc2cizrmnpi...fwxmzl5gjledgonlq
baremetal_core_instance.TFInstance: Still creating... (10s elapsed)
baremetal_core_instance.TFInstance: Still creating... (1m10s elapsed)
baremetal_core_instance.TFInstance: Creation complete (ID: ocid1.instance.oc1.phx.abyhqljrik3gbkov...joifwh2x75xtn
data.baremetal_core_vnic_attachments.InstanceVnics: Refreshing state...
baremetal_core_volume_attachment.TFBlock0Attach: Creating...
  volume_id:           "" => "ocid1.volume.oc1.phx.abyhqljrc2cizrmnpimw54fwxmzl5gjledgonlqqoka2ndrpsx3f2byqjwtq"
data.baremetal_core_vnic.InstanceVnic: Refreshing state...
null_resource.remote-exec: Creating...
null_resource.remote-exec: Provisioning with 'remote-exec'...
null_resource.remote-exec (remote-exec): Connecting to remote host via SSH...
baremetal_core_volume_attachment.TFBlock0Attach: Creation complete (ID: ocid1.volumeattachment.oc1.phx.abyhqljr...3
null_resource.remote-exec: Creation complete (ID: 2958043854685587895)
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
Outputs:
InstancePrivateIP = [  10.0.0.17  ]
InstancePublicIP = [   129.146.20.219 ]
```

Destroy

- When infrastructure needs to be retired, destroying it and all of its dependencies is straightforward with

```
$ terraform destroy
```

- Terraform destroy will ask for permission , requiring an explicit “yes” as input. Terraform when destroying an infrastructure is very thorough.
- The iterative plan, apply, destroy cycle is useful when learning terraform.
- If a resource is change or removed in the .tf file, the state file will detect this and change or remove the resource on the next apply.
- Tainting may also be used to force the recreation of a resource. There are also lifecycle directives available to protect resources if needed.

```
$ terraform plan -destroy
```

- Shows what will be destroyed without actually doing it.

ORACLE

DIY - Terraform Kubernetes Installer for OCI

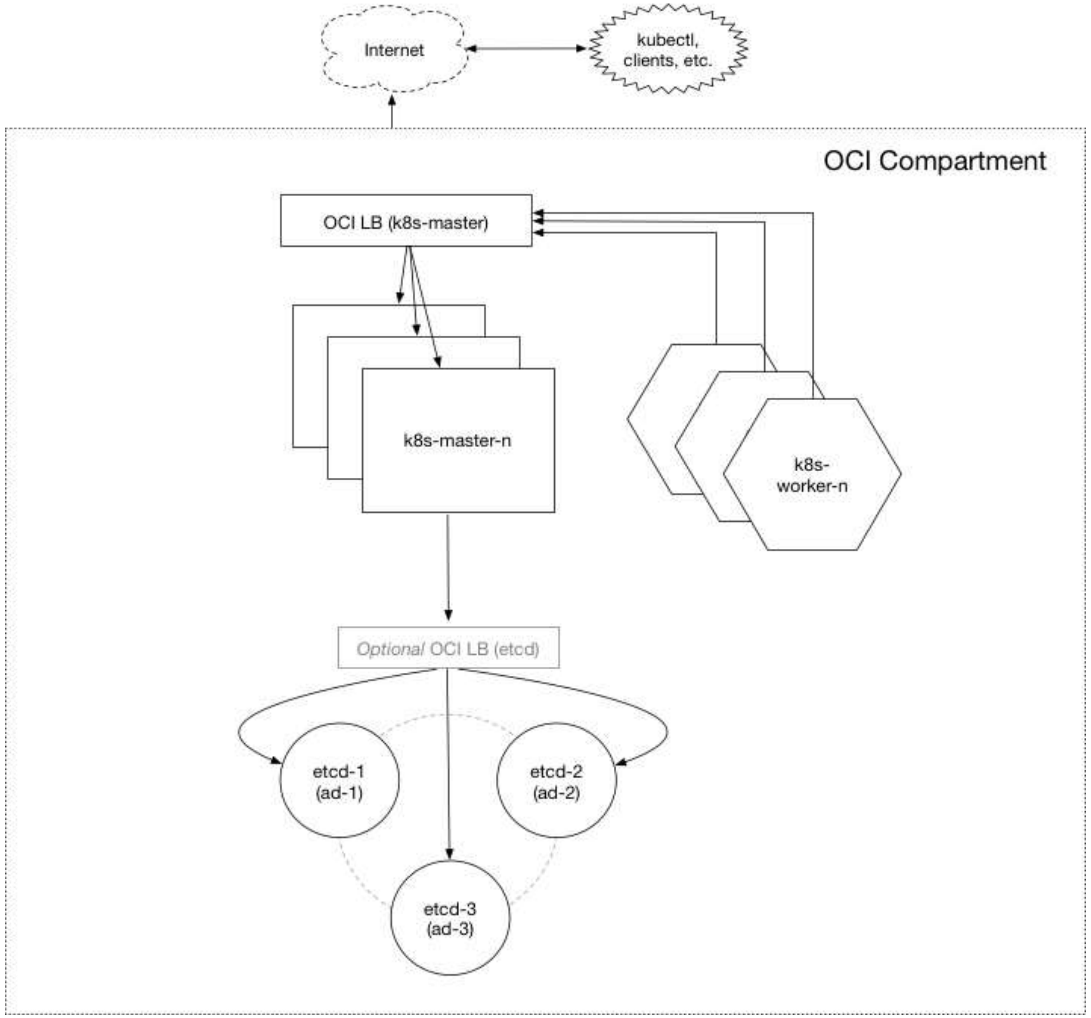
Open Source OCI Kubernetes installer, based on Terraform

- Oracle developed for Kubernetes on OCI
- Available now on Github - https://github.com/oracle/terraform-kubernetes-installer

Key Highlights

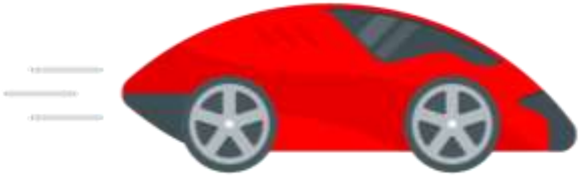
- Highly available Kubernetes cluster configured in your OCI tenancy and compartment
- Creates VCN, subnets, LBs and instances for control plane
- Specify number and shape of nodes for your cluster
- Scale your cluster as needed

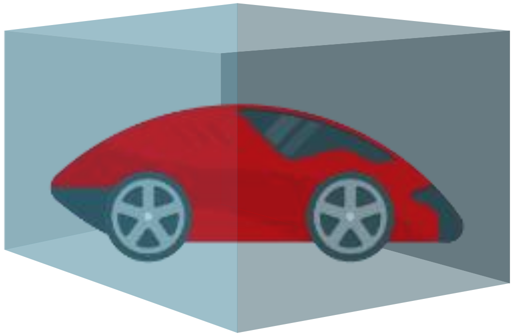
https://blogs.oracle.com/developers/get-a-highly-available-kubernetes-cluster-on-oracle-cloud-infrastructure-in-minutes

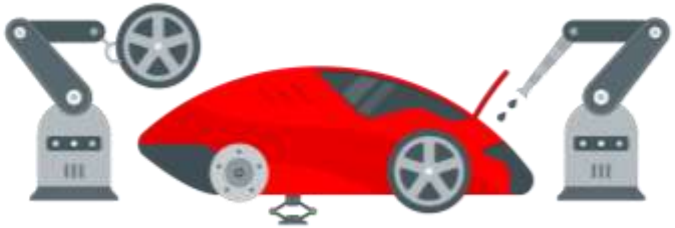


Available on Oracle Github!

Foundational Autonomous Capabilities







Self-Driving

Automates all management, scaling, monitoring, tuning

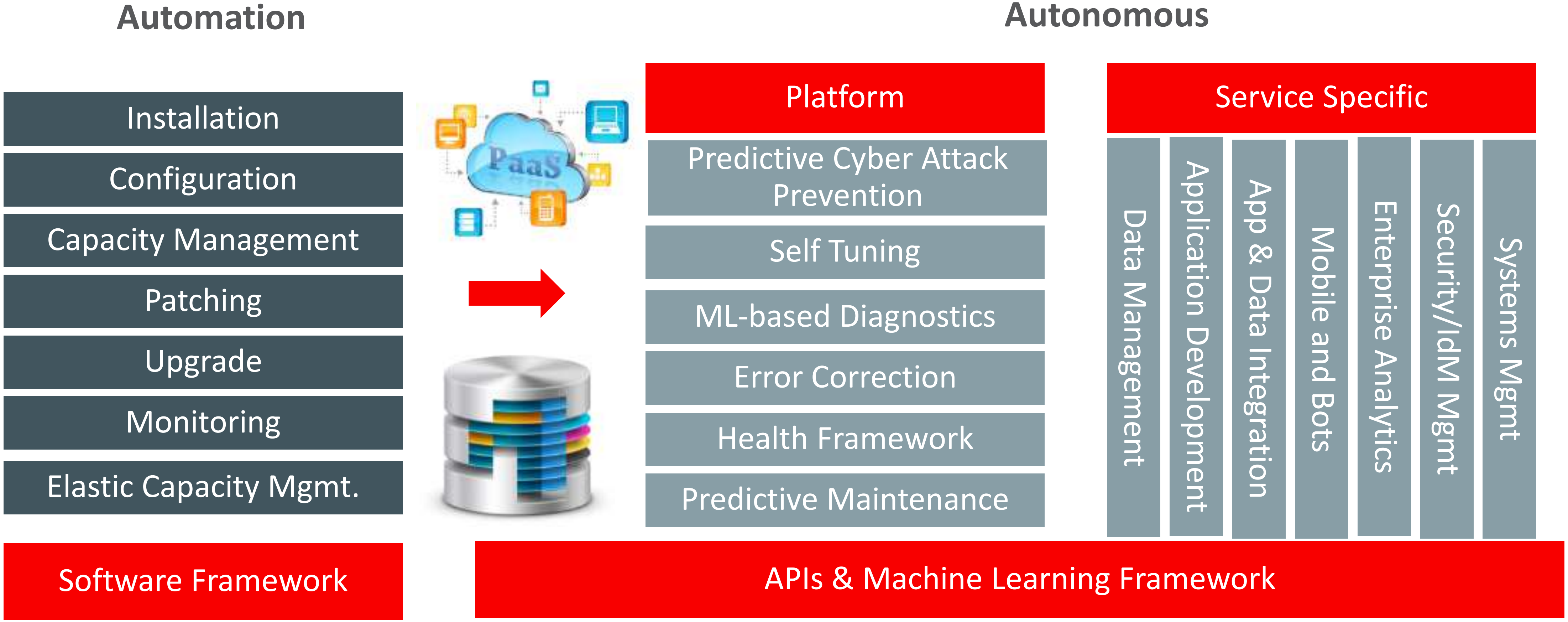
Self-Securing

Protects from both external attacks and malicious internal users

Self-Repairing

Protects from all downtime including planned maintenance

Oracle Cloud Platform Autonomous Service Evolution



Automation

Installation

Configuration

Capacity Management

Patching

Upgrade

Monitoring

Elastic Capacity Mgmt.

Software Framework

Autonomous

Platform

Predictive Cyber Attack Prevention

Self Tuning

ML-based Diagnostics

Error Correction

Health Framework

Predictive Maintenance

Service Specific

Data Management

Application Development

App & Data Integration

Mobile and Bots

Enterprise Analytics

Security/IdM Mgmt

Systems Mgmt

APIs & Machine Learning Framework

Q&A

ORACLE