U.S. NAVAL RESEARCH LABORATORY

Adventures in the Cloud

(or *How I learned to stop worrying and love elasticity*)

Tim Whitcomb

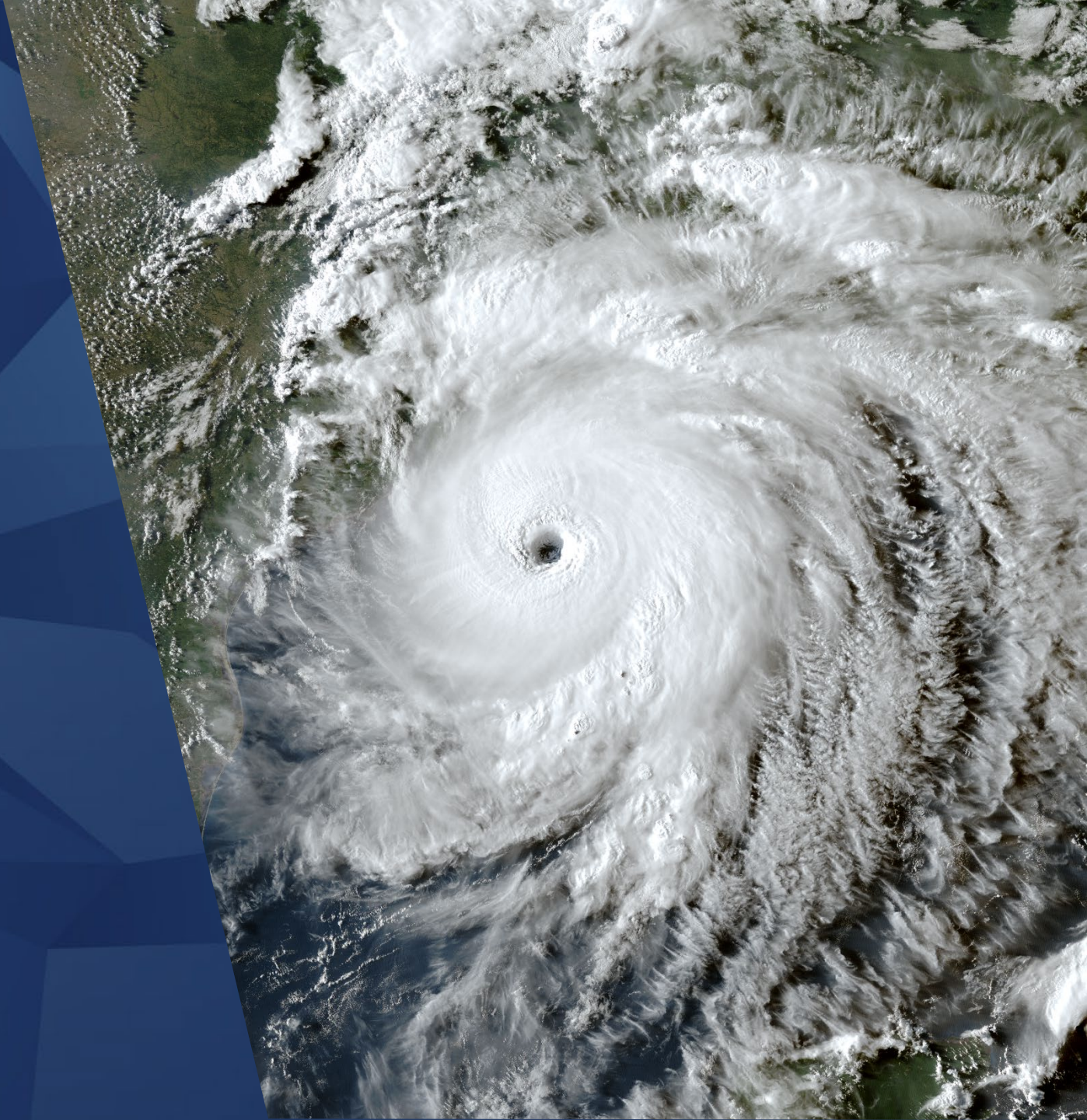
NRL Marine Meteorology Division, Monterey, CA

October 2022 ICAP Meeting

Conclusions

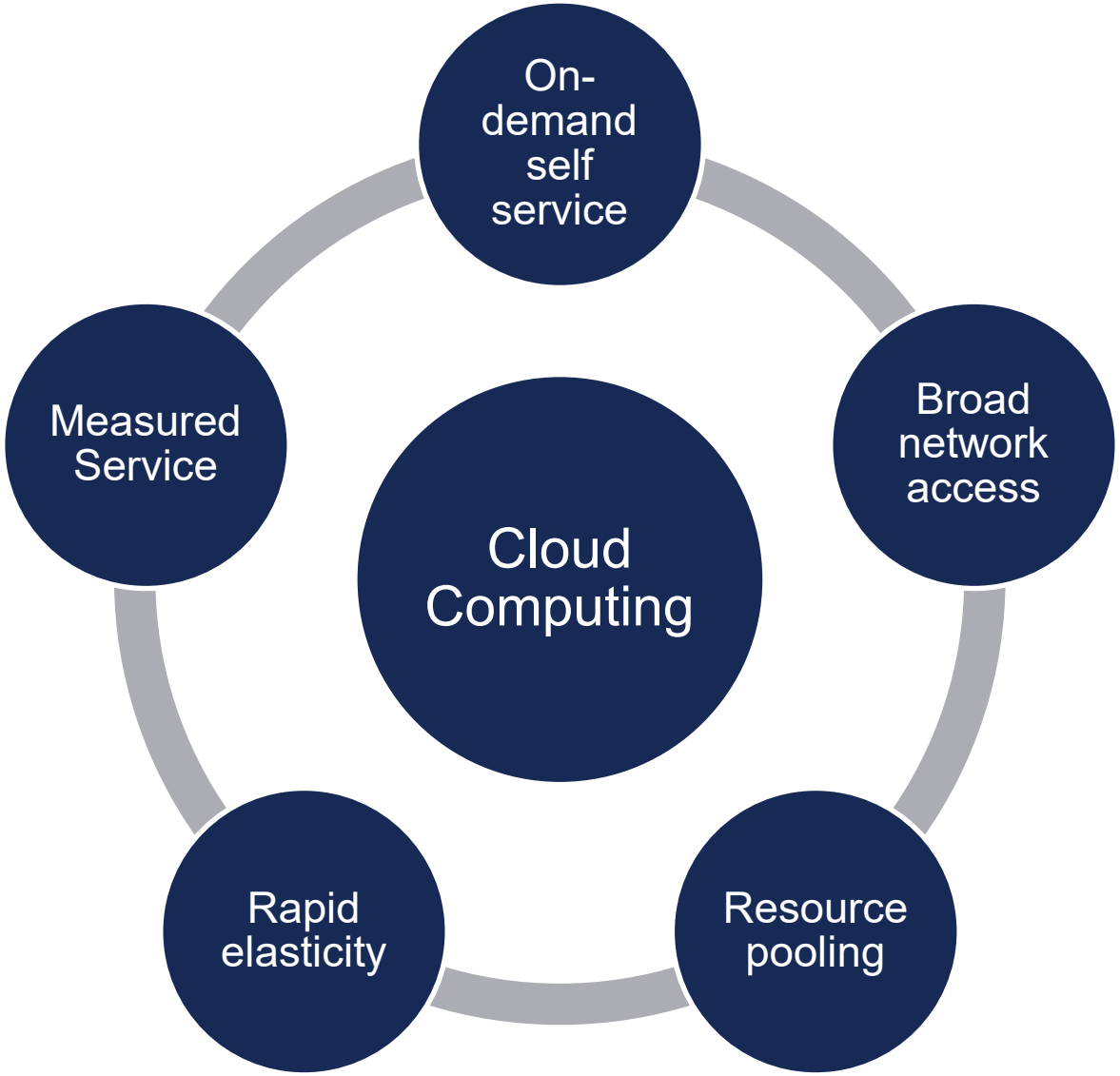
1. Leverage PaaS to accelerate experimentation
2. Treat *everything* as code
3. Containers solve problems
4. Architect for elasticity

What is cloud computing anyway?



U.S. NAVAL RESEARCH LABORATORY

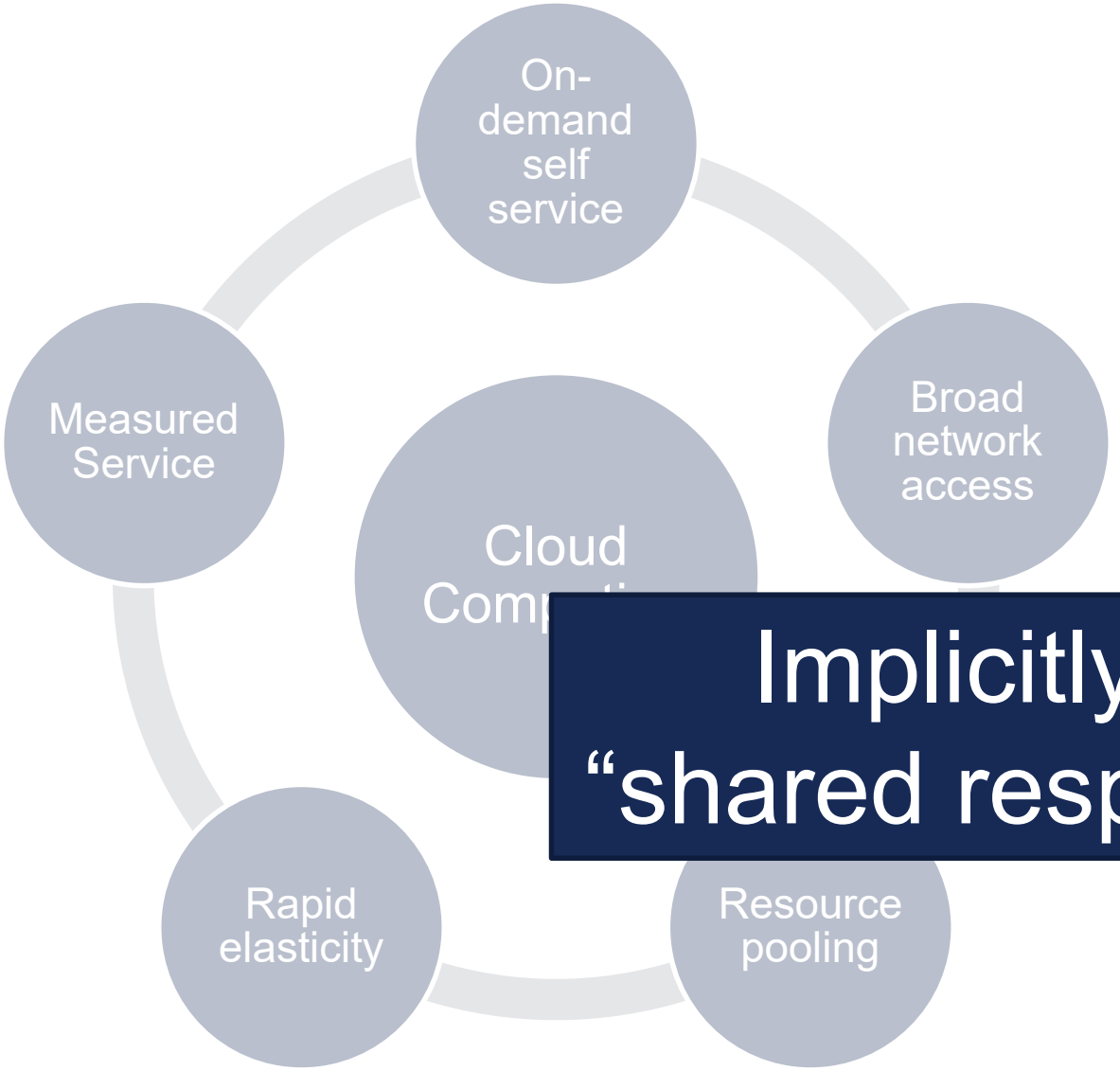
Cloud Computing



"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

- The NIST Definition of Cloud Computing (Mell & Grance, 2011)

Cloud Computing



"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider

d Computing (Mell

Implicitly includes the "shared responsibility model"

Leverage PaaS to accelerate experimentation

U.S. NAVAL RESEARCH LABORATORY

As-a-Service Spectrum

Category	What is it?	IaaS	PaaS	SaaS
Infrastructure	Networks, virtualization, hardware	Vendor	Vendor	Vendor
Platform	OS, middleware, API, runtimes	You	Vendor	Vendor
Software	Data and apps	You	You	Vendor

IaaS: I'm deploying VMs and putting my own os and software on them
PaaS: I'm calling APIs for vendor products to build my software
SaaS: I'm using a pre-existing software package

(adapted from https://blog.hubspot.com/service/iaas-paas-saas)

Case Study: TC Ensemble Forecasting

Requirement: A method to pass event data between distributed actors (i.e. storage, forecast workflow orchestration, dissemination workflows)

Solution: Something like Apache Kafka is fit-for-purpose to support the requirement:

- Supports publish/subscribe messaging
- Supports multiple publishers
- Supports multiple subscribers
- Well-supported client API

Case Study: TC Ensemble Forecasting

Requirement: A method to pass event data between distributed actors (i.e. storage, forecast workflow orchestration, dissemination workflows)

Challenge: If you want to use Kafka, you're going to have to provision a VM, install an OS, install Kafka, configure Kafka, and configure clients

- Supports multiple publishers
- Supports multiple subscribers
- Well-supported client API

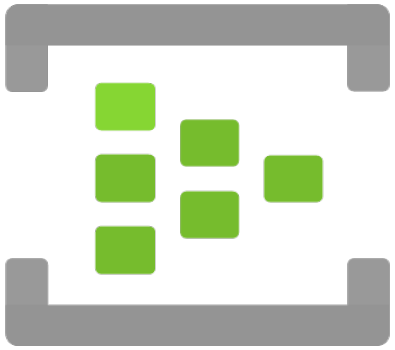
This is the IaaS approach

Case Study: TC Ensemble Forecasting

Requirement: A method to pass event data between distributed actors (i.e. storage, forecast workflow orchestration, dissemination workflows)

Solution: Use Azure Event Hubs as a PaaS alternative to Kafka

- Skip VMs, OS, etc. right to “client configuration”
- Use to test out and prove the design
- *Accelerate* development of a working prototype



Case Study: TC Ensemble Forecasting

Requirement: A method to pass event data between distributed actors (i.e. storage, forecast workflow orchestration, dissemination workflows)

Solution: Use Azure Event Hubs as a PaaS alternative to Kafka

- Skip VMs, OS, etc. right to “client configuration”
- Use to test out and prove the design
- *Accelerate* development of a working prototype

There are good reasons (especially long-term) to use IaaS and Kafka. Starting with PaaS reduces the activation energy of new ideas and new designs

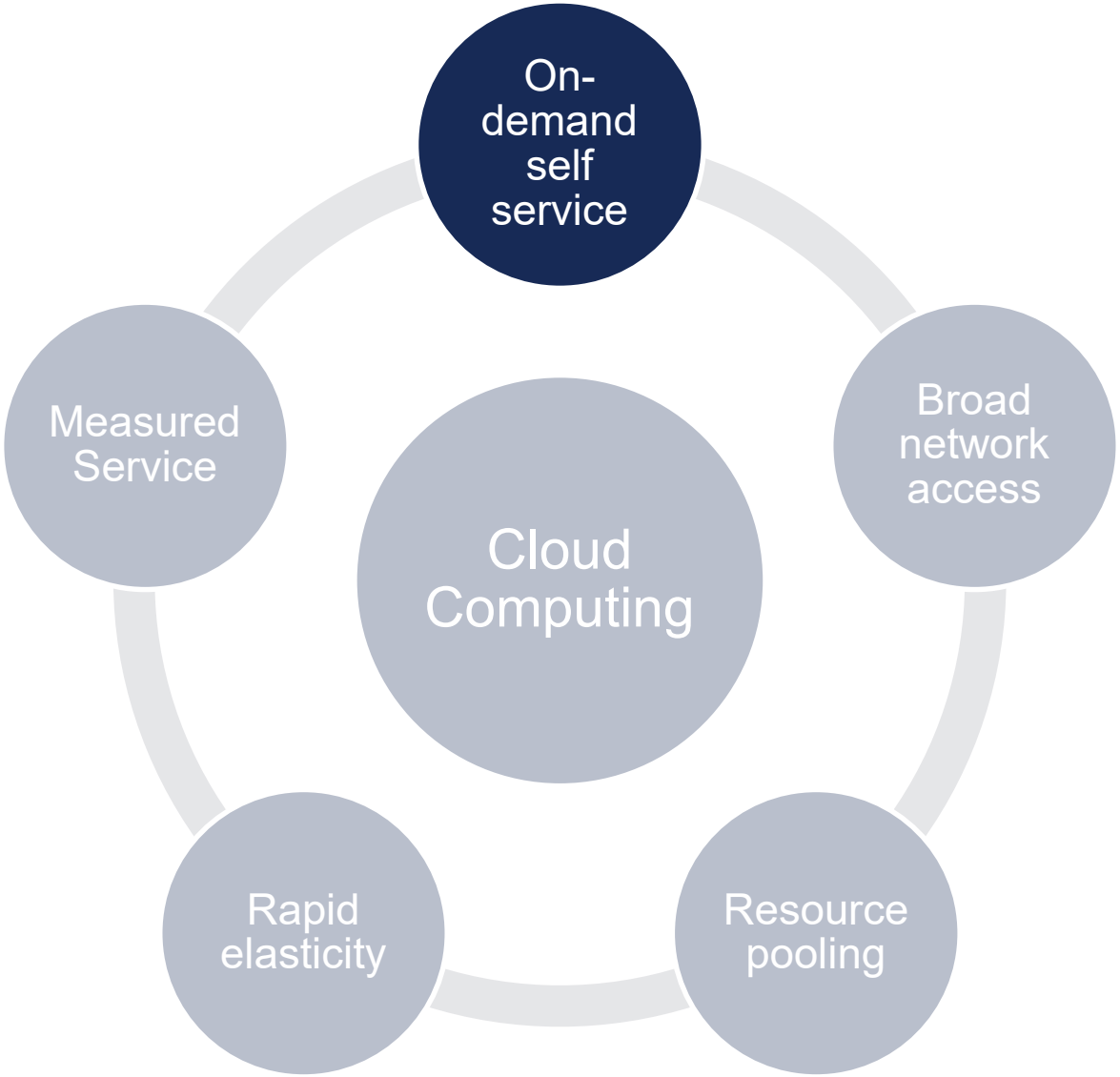
Treat Everything as Code

U.S. NAVAL RESEARCH LABORATORY

Code all the way down

- Applications already consist of instructions in code
  - But they need to be built and tested
- Continuous integration pipeline tools use code to specify automated build and test
  - But they need to run on an OS
- Provisioning tools like Ansible, Chef, Puppet use code to configure systems (e.g. install OS updates, install software stack, apply security patches)
  - But they need to run on hardware/virtualized platforms

Cloud Computing



Cloud resources are deployed via an application programming interface (API) – so cloud deployments are a *sequence of API calls*

Which gives rise to **Infrastructure as Code**

Case Study: Ship Routing

Scenario: Need to repeatedly and reliably deploy cloud resources to support development and demonstration for ONR 6.2 ship routing project.

Solution: Capture deployment configuration (from networks to web services) in Terraform script

Capturing the deployment in Terraform provided confidence to tear down the demo environment and bring it back up with beefier servers

Case Study: Ship Routing

Scenario: Need to repeatedly and reliably deploy cloud resources to support development and demonstration for ONR 6.2 ship routing project.

Solution: Capture deployment configuration (from networks to web services) in Terraform script

Using IaC might slow initial deployments – the *first* time.
Benefits far outweigh the additional cost up front.

Case study: Ship Routing

```
resource "azurerm_virtual_network" "vnet" {
    name = "vnet-${var.project}-${var.environment}-${var.location}"
    location = var.location
    resource_group_name = azurerm_resource_group.resource_group.name
    address_space = ["10.1.0.0/16"]
}

resource "azurerm_subnet" "k8s_nodes_subnet" {
    name = "snet-k8s-${var.project}-${var.environment}-${var.location}"
    resource_group_name = azurerm_resource_group.resource_group.name
    virtual_network_name = azurerm_virtual_network.vnet.name
    address_prefixes = ["10.1.2.0/24"]
}
```

IaC isn't just for storage or virtual machines – all the way down to the *network architecture* that is used.

Containers Solve Problems

(even problems you didn't know you had!)

U.S. NAVAL RESEARCH LABORATORY

Containers Drive Portability

"I don't know why it doesn't work on my on-prem system – it works fine at DSRC"

"I need to update to a new library version, but it'll break that other code"

"Are you sure you're running exactly what we gave you?"

Containers Drive Portability

"We moved from Koehr to Narwhal and things still ran"

"I need to issue a new image with updated libraries for the new feature."

"This runs on-premises and in the cloud"

Containers Drive Portability

We moved from Koehr to Narwhal and things still ran

I need to issue a new image with updated libraries for the new

Containers package an application and its dependencies: all the host needs to provide is a container runtime.

This runs on-premises and in the cloud

Containers Drive Portability

We moved from Koehr to Narwhal and things still ran

I need to issue a new
n updated
or the new
feature!

This runs on-premises and in the cloud

Containers will help make your implicit dependencies explicit!

Case Study: Model Evaluation Toolkit

- Model Evaluation Toolkit developed by NCAR Developmental Testbed Center (DTC)
- Community package for forecast verification tasks – traditional metrics, object-based methods, and more
- Long list of dependencies: BUFRLIB, NetCDF4, Cairo, GRIB2C, etc.
- Non-trivial installation effort

Case Study: Model Evaluation Toolkit

Configure the simplest Singularity recipe:

```
Bootstrap: docker
From: dtcenter/met:8.1
```

Built on NRL workstation

Published Singularity image to Navy DSRC HPC (a file copy!)

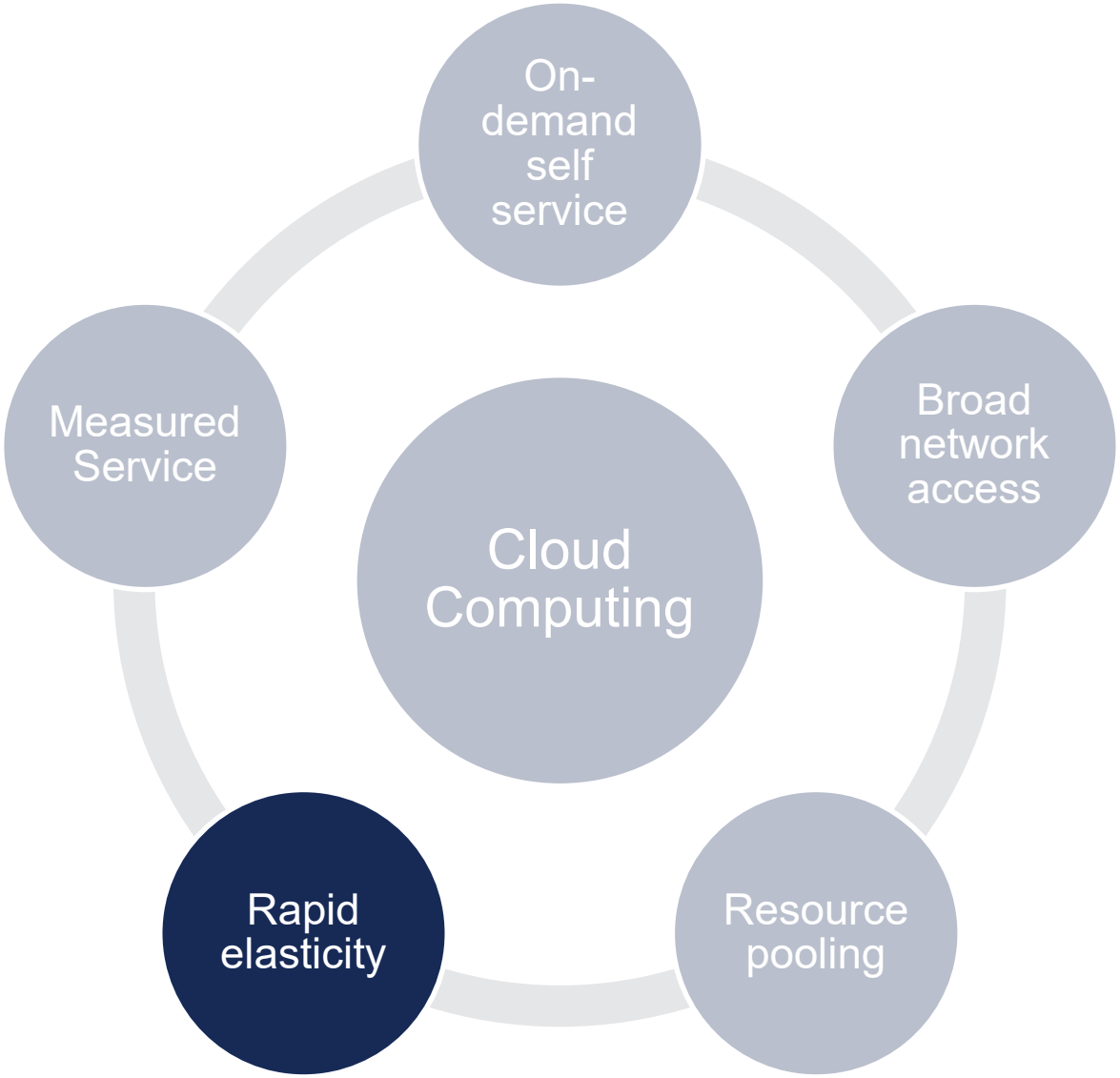
Executed MET via Singularity image at Navy DSRC

No additional new stuff required!

Architect for Elasticity

U.S. NAVAL RESEARCH LABORATORY

Cloud Computing



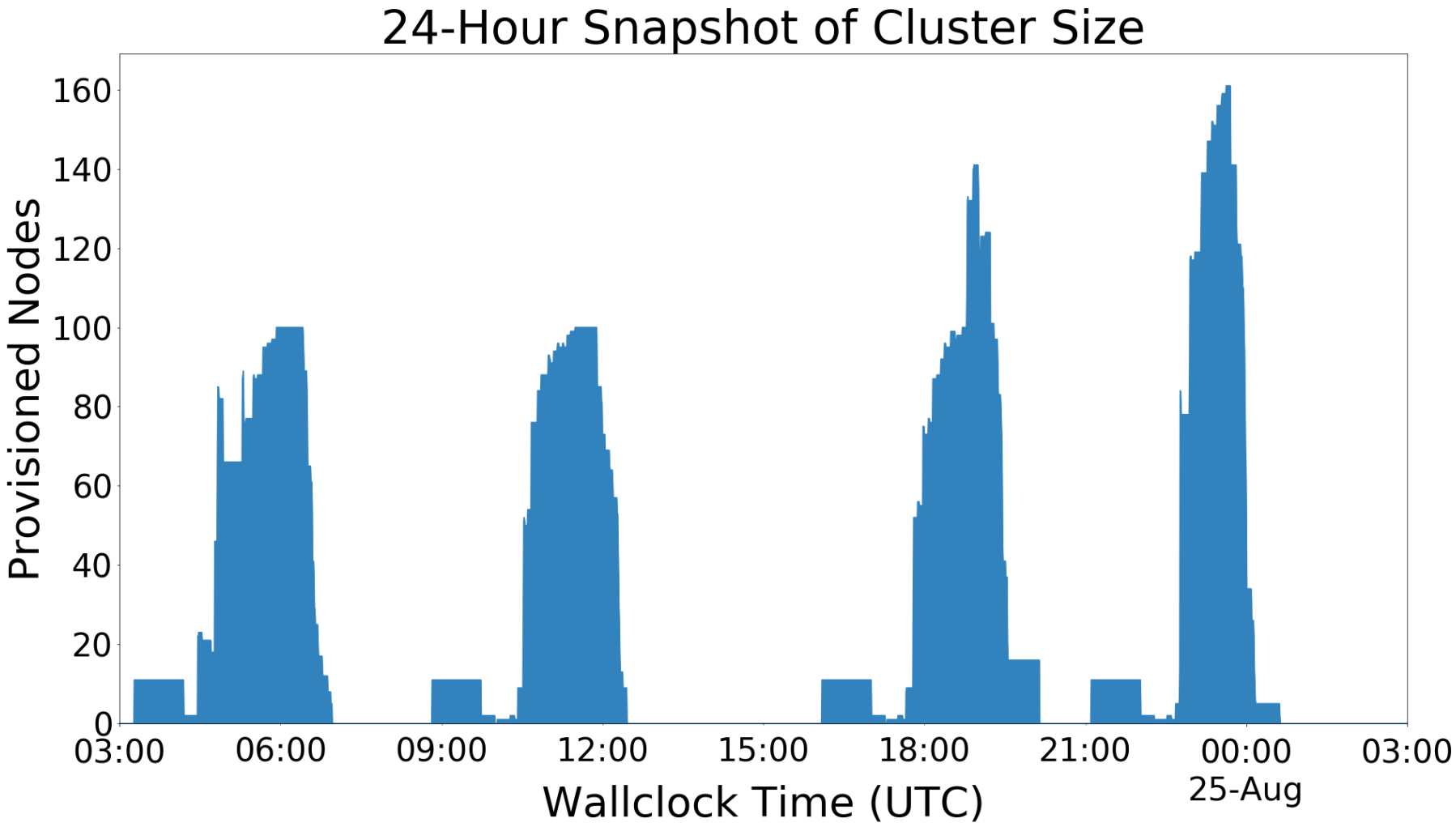
Provision things when you need them and delete them when you don't.

Case Study: TC Ensemble Forecasting

Requirement: Execute a 21-member forecast ensemble for every active tropical cyclone identified by NHC & JTWC optimizing for time-to-solution.

Solution: A system that looks at first glance like an on-premises HPC system, but that integrates with the PBS job scheduler to spin up new compute nodes when the queue starts to back up.

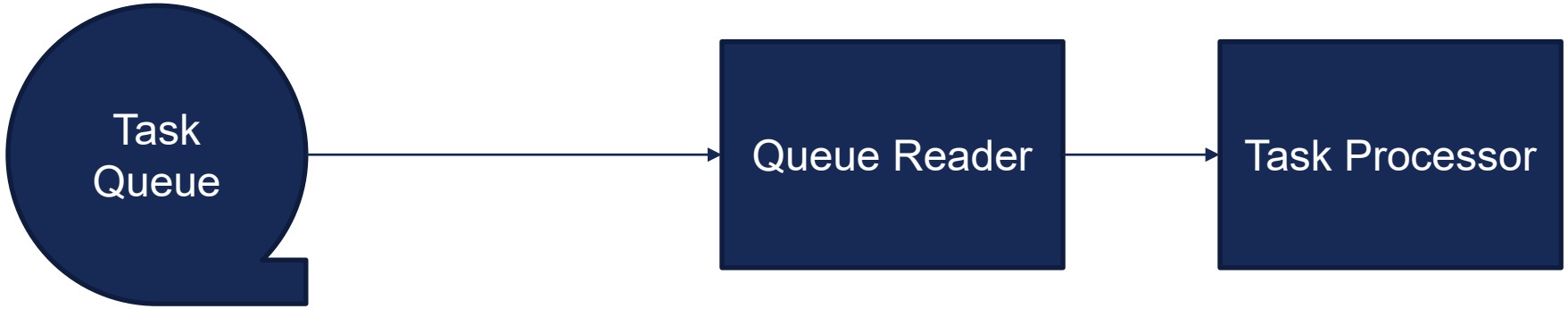
Case Study: TC ensemble Forecasting



(Fig. 2 from Whitcomb, et al. 2022)

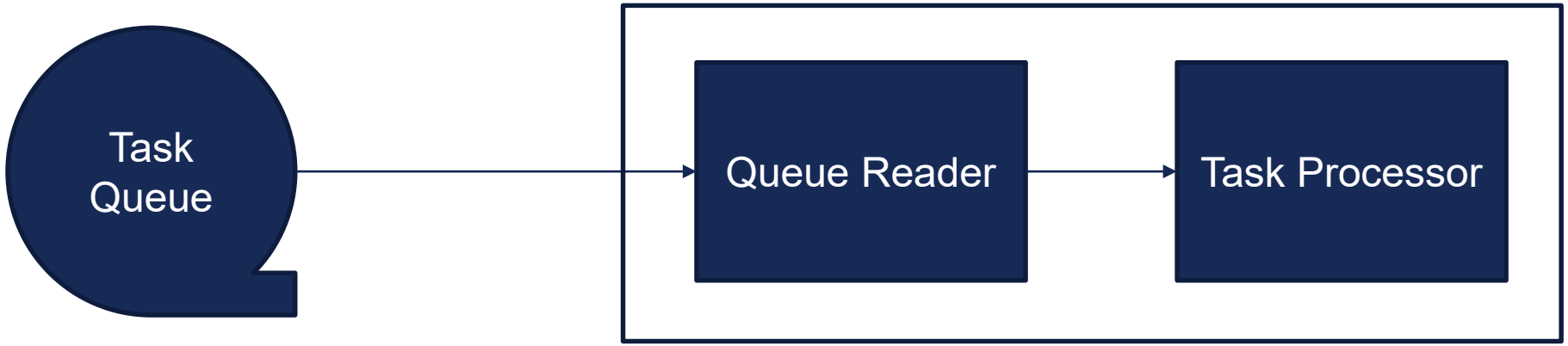
U.S. NAVAL RESEARCH LABORATORY

Case Study: METOC Impact Calculations



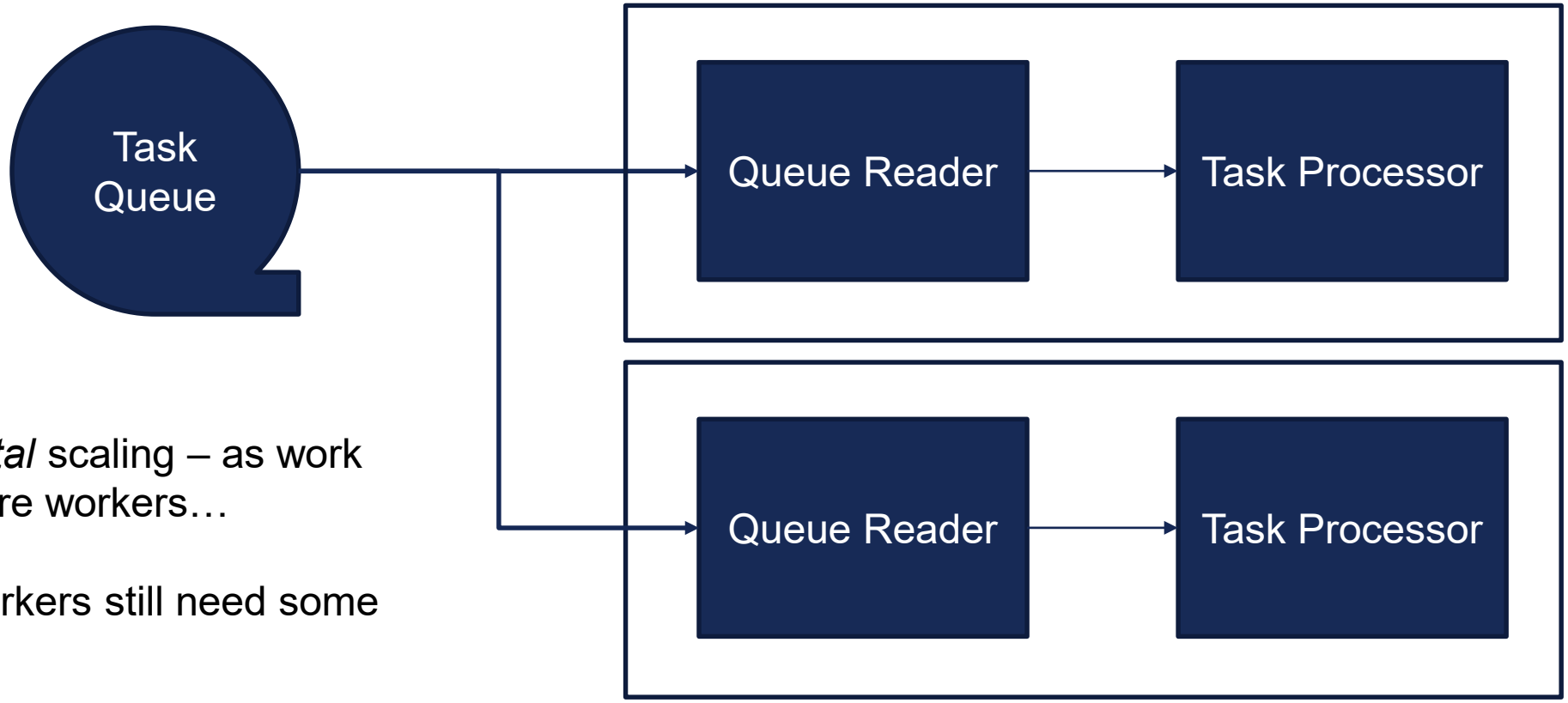
Simple process with simple components – grab a piece of work off a queue, and run it through a standalone application

Case Study: METOC Impact Calculations



Architecting for elasticity starts by identifying *scalable components*

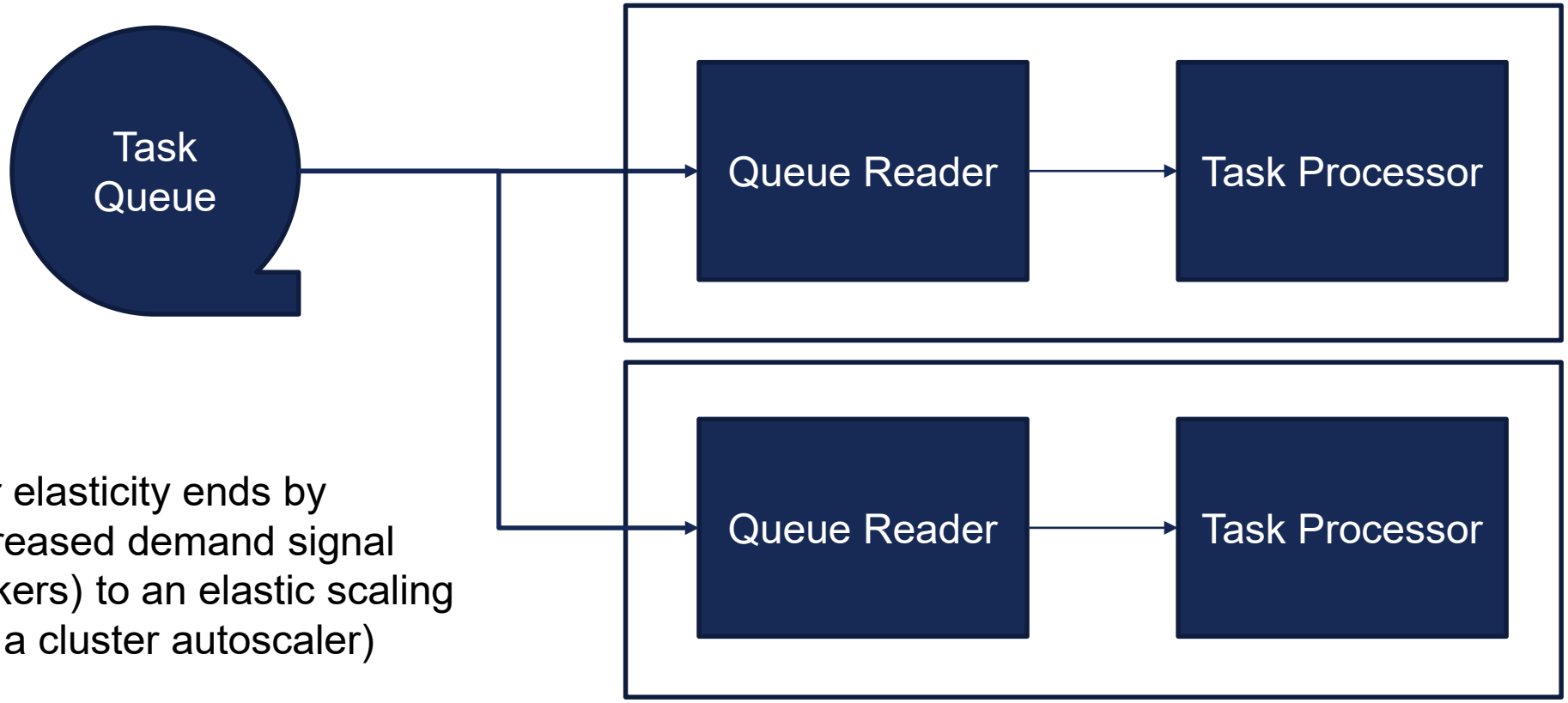
Case Study: METOC Impact Calculations



This is *horizontal* scaling – as work grows, add more workers…

…but those workers still need some place to run!

Case Study: METOC Impact Calculations



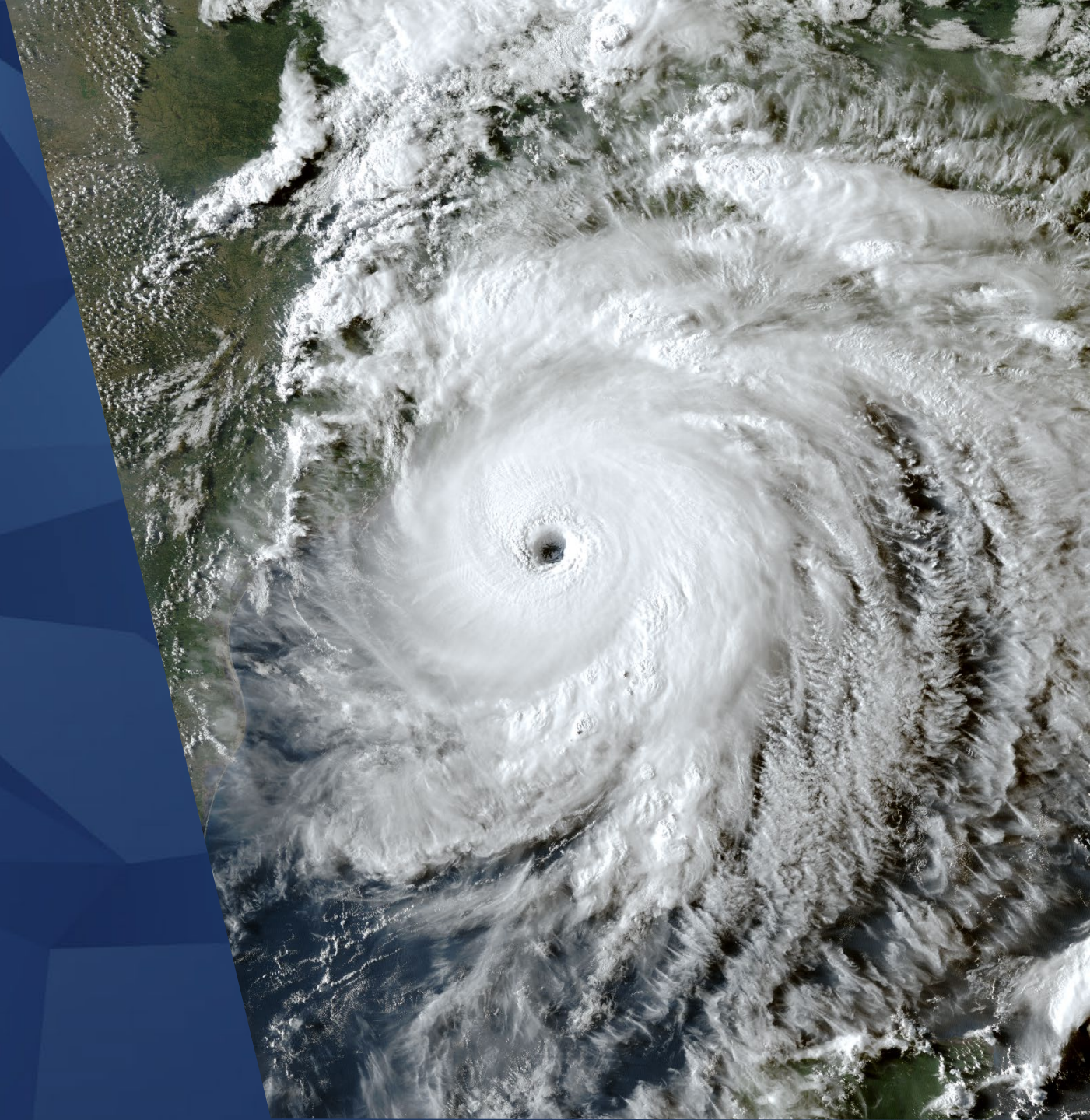
Architecting for elasticity ends by connecting increased demand signal (e.g. more workers) to an elastic scaling capability (e.g. a cluster autoscaler)

Case Study: METOC Impact Calculations

Architecting f
connecting in
(e.g. more wo
capability (e.g

Successful elastic architectures rely on implementing multiple best practices like containers and automated infrastructure provisioning

Conclusions



U.S. NAVAL RESEARCH LABORATORY

1. **Leverage PaaS to accelerate experimentation**
2. **Treat *everything* as code**
3. **Containers solve problems**
4. **Architect for elasticity**

U.S. NAVAL RESEARCH LABORATORY

1. **Leverage PaaS to accelerate experimentation**
2. **Treat *everything* as code**
3. **Containers solve problems**
4. **Architect for elasticity**

Familiarity with cloud computing allows me to **think differently** about problem solving

U.S. NAVAL RESEARCH LABORATORY