AI VOID FIELD GUIDE

Step-by-Step Tutorials

Focused, practical tutorials that teach you how to accomplish specific tasks step by step. Learn by doing with clear instructions and working code examples.

AI VOID

2026-06-23 15:32 IST

Contents

CHAPTER 01

Automate OCI Benchmarks with Terraform

What you'll build: A Terraform stack deployed via OCI Resource Manager to automate the provisioning of compute and storage resources, execute benchmarks (Sysbench, FIO), and collect results on Oracle Cloud Infrastructure. **Time needed:** ~75 minutes **Prerequisites:** An active Oracle Cloud Infrastructure (OCI) account with administrative privileges, Basic understanding of Terraform concepts and HCL syntax, OCI CLI installed and configured for authentication, SSH key pair for accessing compute instances **Version scope:** Terraform: v1.x (stable); OCI Terraform Provider: 8.19.0; OCI Resource Manager: official-docs-current **Last verified:** 2026-06-23 against official docs (https://registry.terraform.io/providers/oracle/oci/latest/docs)

Benchmarking cloud infrastructure is crucial for making informed decisions about resource sizing, cost optimization, and performance tuning. Manually provisioning resources, installing tools, running tests, and collecting data can be a time-consuming and error-prone process. This tutorial empowers you to automate this entire workflow on Oracle Cloud Infrastructure (OCI) using Terraform and OCI Resource Manager, ensuring consistent, repeatable, and efficient benchmarking.

By the end of this guide, you'll have a robust, automated system for evaluating OCI compute and storage performance, helping you understand how different configurations impact your applications.

Prerequisites and OCI Account Setup

Before we dive into automating benchmarks, let's ensure your OCI environment and local workstation are ready. This setup is foundational for interacting with OCI programmatically.

1. Active OCI Account with Administrative Privileges

You'll need an active OCI account. For this tutorial, we assume you have administrative privileges (or a user with permissions to create/manage Compute instances, Block Volumes, Networking, and Resource Manager stacks). If you're new to OCI, you can sign up for a Free Tier account to get started.

2. OCI CLI Installation and Configuration

The OCI Command Line Interface (CLI) is essential for authenticating Terraform with your OCI account and for performing various OCI operations.

First, ensure you have the OCI CLI installed. If not, follow the official installation guide for your operating system.

Next, you need to configure the OCI CLI for authentication. This typically involves generating an API signing key pair and configuring the `~/.oci/config` file.

Open your terminal and run the configuration command:

```
oci setup config
```

This command will guide you through creating an API key pair and setting up your OCI configuration file.

> ⚡ Note: This process usually prompts you for your User OCID, Tenancy OCID, Region, and the path to your API key. Make sure to keep your private key secure.

What to verify: After running `oci setup config`, you should have a `~/.oci/config` file and a private key file (e.g., `~/.oci/oci_api_key.pem`) in place. You can test your configuration by running a simple OCI CLI command, like listing your compartments:

```
oci iam compartment list --all
```

If this command returns a list of compartments without errors, your OCI CLI is correctly configured.

3. SSH Key Pair for Compute Instance Access

The benchmark automation stack will provision OCI Compute instances. To access these instances later to inspect logs or manually run commands (if needed), you'll need an SSH key pair.

If you don't already have one, you can generate a new SSH key pair using `ssh-keygen`:

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/oci_benchmark_key
```

This command generates a private key (`oci_benchmark_key`) and a public key (`oci_benchmark_key.pub`) in your `~/.ssh` directory.

> ⚠️ Common mistake: When generating SSH keys, ensure you save them in a secure location and provide the public key to Terraform/Resource Manager. Never expose your private key.

What to verify: You should have two files: `~/.ssh/oci_benchmark_key` (private key) and `~/.ssh/oci_benchmark_key.pub` (public key). You'll need the content of the public key for the Terraform configuration.

Section Accomplishment

You have successfully prepared your OCI account and local environment by configuring the OCI CLI and generating an SSH key pair, laying the groundwork for automated deployments.

Understanding the Benchmark Automation Terraform Stack

At the heart of this tutorial is a pre-built Terraform stack designed to automate OCI compute and storage benchmarking. This stack simplifies a complex process into a single, declarative configuration.

What is this Terraform Stack?

This is an open-source Terraform configuration that defines the OCI resources needed for benchmarking and orchestrates the execution of benchmark tools. It's designed to be deployed via OCI Resource Manager, Oracle's managed Terraform service.

Key components managed by the stack:

- **Networking:** A Virtual Cloud Network (VCN), subnets, and security lists to isolate and secure your benchmark environment.
- **Compute Instances:** One or more OCI Compute instances (VMs) where the benchmarks will run. You can specify instance shape, OCI image, and other properties.
- **Block Volumes:** Attached and configured OCI Block Volumes to test storage performance. The stack handles attachment, formatting, and mounting.

- **Benchmark Tools:** Installation and execution of `Sysbench` (for CPU, memory, and I/O) and `FIO` (for detailed storage I/O).
- **Scripting:** User data scripts to automate the installation of benchmark tools, execution of tests, and collection of results.

> 📌 Key Idea: Using a pre-built Terraform stack ensures consistency. Every time you deploy it, you get the exact same environment and benchmark setup, minimizing variables that could affect results.

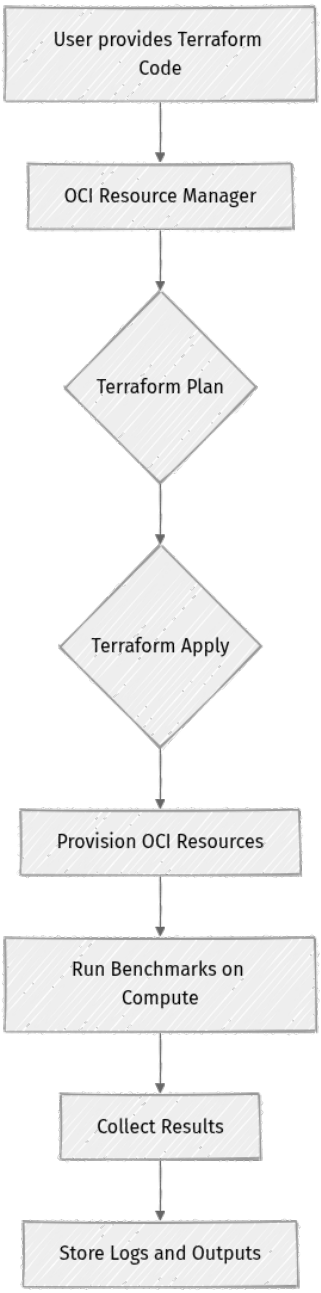
How Resource Manager Fits In

OCI Resource Manager is a managed service that allows you to deploy, manage, and tear down OCI resources using Terraform configurations. Instead of running `terraform` commands from your local machine, you upload your Terraform code to Resource Manager, and it handles the execution.

Benefits of using Resource Manager for this task:

- **Centralized State Management:** Resource Manager stores your Terraform state in OCI, making it accessible to team members and preventing local state inconsistencies.
- **Execution Environment:** It provides a consistent and reliable environment to run Terraform operations, abstracting away local Terraform CLI installations and dependency issues.
- **Version Control Integration:** You can link Resource Manager stacks directly to Git repositories, allowing for version-controlled infrastructure as code.
- **Auditing and Logging:** All operations performed by Resource Manager are logged, providing an audit trail of infrastructure changes.

Here's a simplified flow of how the components interact:



Section Accomplishment

You now understand the purpose and components of the benchmark automation Terraform stack and why OCI Resource Manager is the ideal platform for deploying it, providing a robust, managed environment.

Configuring OCI for Resource Manager Deployment

Before we upload our Terraform stack, we need to prepare our OCI tenancy with a few essential components: a compartment for organization, and an SSH key that Resource Manager can use to configure the compute instances.

1. Create a Dedicated Compartment

It's a best practice to organize your OCI resources into compartments. This helps with access control, billing, and resource isolation. Let's create a new compartment specifically for our benchmark resources.

1. Log in to the OCI Console.
2. Navigate to **Identity & Security** > **Compartments**.
3. Click **Create Compartment**.
4. Enter the following details:
    - **Name:** `BenchmarkAutomation` (or a name of your choice)
    - **Description:** `Compartment for OCI compute and storage benchmark automation.`
    - **Parent Compartment:** Choose your root compartment or a suitable parent.
5. Click **Create Compartment**.

What to verify: Once created, navigate to the `BenchmarkAutomation` compartment. Note its OCID, as you might need it later for policies or other configurations, though Resource Manager typically uses the compartment name.

2. Upload SSH Public Key to OCI

The Terraform stack will need to inject your SSH public key into the provisioned compute instances. While you could provide the public key directly in the Terraform variables, a more secure and reusable approach is to upload it as an OCI `ssh_key` resource.

1. In the OCI Console, navigate to **Compute** > **SSH Keys**.
2. Click **Upload SSH Key**.
3. Choose the compartment you just created (`BenchmarkAutomation`).
4. Give the key a **Name**, e.g., `oci_benchmark_key`.
5. Paste the contents of your public key file (`~/.ssh/oci_benchmark_key.pub`) into the **SSH Key Value** field.
    - To get the content of your public key:

```
cat ~/.ssh/oci_benchmark_key.pub
```

1. Click **Upload SSH Key**.

> ⚡ Note: This step is crucial. The Terraform configuration will reference this uploaded SSH key by its name, allowing the compute instances to be provisioned with your public key for secure access.

What to verify: After uploading, you should see your SSH key listed under **SSH Keys** within your `BenchmarkAutomation` compartment.

Section Accomplishment

You have successfully prepared your OCI tenancy by creating a dedicated compartment for your benchmark resources and uploading your SSH public key, ensuring proper organization and secure access for the deployed compute instances.

Creating and Deploying the Resource Manager Stack

Now that our OCI environment is ready, let's get the benchmark automation Terraform configuration and deploy it using OCI Resource Manager.

1. Obtain the Terraform Configuration

The benchmark automation stack is available as an open-source project. You'll typically find it on GitHub. For this tutorial, we'll assume a structure similar to the official Oracle blog post reference.

You can download the Terraform configuration files as a ZIP archive.

1. Navigate to the official Oracle blog post describing this solution.
2. Look for a link to the GitHub repository or a direct download of the Terraform stack. Assuming it's a repository, you'd clone it or download the ZIP.
    - If cloning:

```
git clone https://github.com/oracle-samples/oci-compute-storage-benchmark-automation.git
cd oci-compute-storage-benchmark-automation
```

```
- If downloading ZIP: Download the repository as a ZIP file, then extract its
contents.
```

```
unzip oci-compute-storage-benchmark-automation-main.zip
cd oci-compute-storage-benchmark-automation-main
```

Make sure you have the `.tf` files (e.g., `main.tf`, `variables.tf`, `outputs.tf`) and any associated scripts within a single directory. This directory will be compressed into a ZIP file for Resource Manager.

2. Prepare the Terraform ZIP File

OCI Resource Manager expects a ZIP file containing all your Terraform configuration (`.tf` files and any other necessary scripts).

1. Navigate into the directory containing the Terraform files (e.g., `oci-compute-storage-benchmark-automation-main`).
2. Create a ZIP archive of the contents of this directory.

```
zip -r benchmark_stack.zip ./*
```

> ⚡ Note: Ensure you ZIP the contents of the directory, not the directory itself. The `.tf` files should be at the root of the ZIP archive.

What to verify: You should have a `benchmark_stack.zip` file containing your Terraform configuration.

3. Create a Resource Manager Stack

Now, let's create a new stack in OCI Resource Manager and upload our configuration.

1. In the OCI Console, navigate to **Developer Services** > **Resource Manager** > **Stacks**.
2. Click **Create Stack**.
3. For **Source Code Management**, select **My Configuration**.
4. For **Terraform Configuration Source**, select **.ZIP file**.
5. Click **Browse** and upload your `benchmark_stack.zip` file.

6. Click **Continue**.
7. On the **Stack Information** page:
    - **Name:** `OCI_Benchmark_Stack` (or a descriptive name)
    - **Description:** `Automated OCI compute and storage benchmarking.`
    - **Compartment:** Select your `BenchmarkAutomation` compartment.
8. Click **Next**.
9. On the **Configure Variables** page, you'll see a list of variables defined in the Terraform configuration (`variables.tf`). These are critical for customizing your deployment.
    - **`oci_region`**: Your OCI region (e.g., `us-ashburn-1`).
    - **`compartment_ocid`**: The OCID of your `BenchmarkAutomation` compartment. You can find this by navigating to **Identity & Security** > **Compartments** and clicking on your compartment.
    - **`ssh_public_key_name`**: The name of the SSH key you uploaded earlier (e.g., `oci_benchmark_key`).
    - **`instance_shape`**: The desired OCI Compute instance shape (e.g., `VM.Standard.E4.Flex` or `VM.Standard.A1.Flex` for Ampere ARM instances if available in your tenancy).
    - **`block_volume_size_in_gbs`**: Size of the block volume in GBs (e.g., `50`).
    - **`availability_domain`**: An Availability Domain in your region (e.g., `AD-1`).
    - **`image_ocid`**: The OCID of the OCI image to use for the compute instance. You can find common image OCIDs in the OCI Console under **Compute** > **Instances** > **Create Instance** (choose an image and copy its OCID). A common choice is an Oracle Linux image.
    - Review other variables and adjust as necessary (e.g., VCN CIDR, subnet CIDR).
10. Click **Next**.
11. On the **Review** page, verify your settings.
12. Click **Create**.

What to verify: You should now see your `OCI_Benchmark_Stack` listed in the Resource Manager Stacks page. Its status should be "Active" or "Created".

4. Plan and Apply the Stack

With the stack created, it's time to tell Resource Manager to execute the Terraform plan and apply it.

1. From the **Stack Details** page for `OCI_Benchmark_Stack`, click **Terraform Actions**.
2. Select **Plan**.
    - This initiates a "Plan" job. Resource Manager will run `terraform plan` to determine what changes will be made to your OCI infrastructure.
    - Monitor the job log. Once completed, review the plan output for any unexpected changes. It should show additions of VCN, subnets, compute instance, and block volume.
3. Once the "Plan" job is successful and you've reviewed the output, click **Terraform Actions** again.
4. Select **Apply**.
    - This initiates an "Apply" job. Resource Manager will run `terraform apply` to provision the resources defined in your configuration.
    - This process can take 15-30 minutes as OCI provisions the compute instance, block volume, and runs the initial setup scripts.
    - Monitor the job log for progress.

> ⚠️ Common mistake: If the "Apply" job fails, carefully review the job log in Resource Manager. Common issues include incorrect OCIDs, insufficient permissions, invalid instance shapes, or resource limits.

What to verify: The "Apply" job should complete successfully, and the stack status should change to "Applied" or "Active". You can also navigate to **Compute** > **Instances** and **Storage** > **Block Volumes** in the OCI Console to see the newly provisioned resources.

Section Accomplishment

You have successfully obtained the benchmark automation Terraform configuration, packaged it into a ZIP, created a Resource Manager stack, configured its variables, and deployed it, provisioning all necessary OCI resources for benchmarking.

Running and Monitoring the Benchmarks

Once your Resource Manager stack has successfully applied, the benchmark scripts are automatically executed on your OCI Compute instance as part of the instance's user data. You'll monitor the progress and completion through the Resource Manager job logs.

1. Automatic Benchmark Execution

The Terraform stack is designed to be self-contained. The `cloud-init` scripts (or user data scripts) embedded in the compute instance definition handle the following:

- Installing necessary packages (e.g., `sysbench`, `fio`).
- Configuring the attached block volume (formatting, mounting).
- Executing `sysbench` tests for CPU, memory, and file I/O.
- Executing `fio` tests for block volume I/O.
- Redirecting benchmark output and logs to specific files on the instance.

This execution happens immediately after the instance boots up for the first time.

2. Monitoring Benchmark Progress via Resource Manager Logs

The most straightforward way to monitor the execution of the benchmark scripts is through the Resource Manager job logs. Even though the benchmarks run on the compute instance, the `terraform apply` job in Resource Manager will continue to stream output from the instance's `cloud-init` process until it completes.

1. In the OCI Console, navigate to **Developer Services** > **Resource Manager** > **Stacks**.
2. Click on your `OCI_Benchmark_Stack`.
3. In the **Stack Details** page, scroll down to **Jobs**.
4. Click on the most recent **Apply** job.
5. In the **Job Details** page, click **Logs**.

As the apply job progresses, you'll see output related to the provisioning of resources. Eventually, you'll start seeing output from the `cloud-init` scripts, including:

- Package installation messages.

- Disk formatting and mounting steps.
- Output from `sysbench` and `fio` as they run their tests.

> ⚡ Quick Note: The `apply` job will only finish once the `cloud-init` scripts on the instance have completed their execution. This is how Resource Manager knows the entire provisioning and setup process (including benchmarks) is done.

What to verify: Keep an eye on the logs. You should see distinct sections for `sysbench` and `fio` output. The job will eventually show a "Succeeded" status.

3. (Optional) SSH into the Instance for Real-time Monitoring

If you prefer more granular, real-time monitoring or need to debug, you can SSH directly into the compute instance.

1. In the OCI Console, go to **Compute** > **Instances**.
2. Find the instance provisioned by your `OCI_Benchmark_Stack`. Note its Public IP Address.
3. Open your terminal and SSH into the instance using the private key you generated:

```
    ssh -i ~/.ssh/oci_benchmark_key opc@<PUBLIC_IP_ADDRESS>
```

```
(Replace `<PUBLIC_IP_ADDRESS>` with your instance's public IP).
```

1. Once logged in, you can check the `cloud-init` logs:

```
    tail -f /var/log/cloud-init-output.log
```

```
This will show you the real-time output of the benchmark scripts.
```

> ⚠️ What can go wrong: If you can't SSH, check the security list rules for your VCN/subnet to ensure SSH (port 22) is open from your IP. Also, verify you're using the correct private key and `opc` user.

Section Accomplishment

You have successfully monitored the automated execution of benchmarks, understanding that the entire process, from provisioning to test execution, is managed and reported through OCI Resource Manager job logs.

Interpreting Benchmark Results and Logs

After the Resource Manager "Apply" job completes successfully, your compute instance has finished running the benchmarks. The results are stored on the instance itself. Let's retrieve and understand them.

1. Retrieving Benchmark Results via SSH

The most direct way to get the results is to SSH into the instance and download the log files.

1. SSH into your compute instance using the private key and public IP address (as described in the previous section):

```
ssh -i ~/.ssh/oci_benchmark_key opc@<PUBLIC_IP_ADDRESS>
```

1. Once connected, navigate to the directory where the benchmark results are stored. The specific location depends on the Terraform stack's scripts, but a common pattern is `/var/log` or a dedicated benchmark directory. Let's assume `/var/log/oci_benchmarks/`.

```
cd /var/log/oci_benchmarks/
ls -l
```

```
You should see files like `sysbench_cpu.log`, `sysbench_memory.log`,
`sysbench_fileio.log`, `fio_block_volume.log`, etc.
```

1. You can view the contents of these files directly:

```
cat sysbench_cpu.log | less
```

1. To download them to your local machine, use `scp` from your local terminal (not inside the SSH session):

```
    scp -i ~/.ssh/oci_benchmark_key opc@<PUBLIC_IP_ADDRESS>:/var/log/
oci_benchmarks/* .
```

```
This command copies all benchmark log files from the remote instance to your
current local directory.
```

2. Understanding Sysbench Results

`Sysbench` is a versatile benchmarking tool for evaluating CPU, memory, and file I/O performance.

- **CPU Benchmarks (`sysbench_cpu.log`):**
  - Look for `total time` and `events per second`. Higher `events per second` indicates better CPU performance.
  - `min`, `avg`, `max` `response time` (latency) are also important. Lower values are better.
- **Memory Benchmarks (`sysbench_memory.log`):**
  - Focus on `operations performed` and `total transfer`. Higher throughput (MB/sec or GB/sec) is better.
  - `min`, `avg`, `max` `response time` for memory access.
- **File I/O Benchmarks (`sysbench_fileio.log`):
  - This tests file system performance on the root volume. Look at `operations performed`, `read`, `written` (MB/sec).

Example `sysbench` output snippet:

```
...
Execution time (avg/stddev): 1.0000/0.00
Latency (ms):
     min: 0.00
     avg: 0.00
     max: 0.00
     percentile 95: 0.00
     sum: 0.00
...
```

(Note: Actual Sysbench output is more detailed; this is a simplified example.)

3. Interpreting FIO Results

`FIO` (Flexible I/O Tester) is a powerful tool for detailed block device I/O benchmarking, especially useful for attached block volumes.

- **Block Volume Benchmarks (`fio_block_volume.log`):**
  - **IOPS (Input/Output Operations Per Second):** This is a key metric, especially for transactional workloads. Higher IOPS indicates better performance. Look for `IOPS=` in the read/write sections.
  - **Bandwidth (BW):** Measured in MB/s or GB/s. Important for sequential workloads (e.g., large file transfers). Look for `BW=` in the read/write sections.
  - **Latency:** `lat (usec/msec)` provides average, min, max, and percentile latencies. Lower latency is always better.
  - **I/O Depth:** The number of I/O operations outstanding at any given time.
  - **Block Size:** The size of the I/O requests.

Example `fio` output snippet:

```
...
Run status group 0 (all jobs):
   READ: bw=1024MiB/s (1074MB/s), 1024MiB/s-1024MiB/s, io=10.0GiB (10.7GB),
run=10240-10240msec
  WRITE: bw=512MiB/s (537MB/s), 512MiB/s-512MiB/s, io=5.00GiB (5.37GB),
run=10240-10240msec

Disk stats (read/write):
  dm-0: ios=255998/127999, merge=0/0, ticks=163838/81919, in_queue=245757,
util=100.00%
...
```

(Note: FIO output is very verbose and detailed. Focus on the *`bw`* and *`IOPS`* lines in the *`Run status group`* section.)

> ⚡ Real-world insight: Benchmarking is iterative. Run tests with different OCI instance shapes, block volume performance tiers, and FIO parameters (e.g., block size, I/O depth) to find the optimal configuration for your workload.

Section Accomplishment

You have successfully retrieved the benchmark results from your OCI Compute instance and gained an understanding of how to interpret the key performance metrics reported by Sysbench and FIO.

Cleaning Up OCI Resources

After you've collected and analyzed your benchmark results, it's crucial to clean up the provisioned OCI resources to avoid incurring unnecessary costs. OCI Resource Manager makes this process straightforward and reliable.

1. Destroy the Resource Manager Stack

Destroying the stack tells Resource Manager to run `terraform destroy`, which will deprovision all resources that were created by that stack.

1. In the OCI Console, navigate to **Developer Services** > **Resource Manager** > **Stacks**.
2. Click on your `OCI_Benchmark_Stack`.
3. From the **Stack Details** page, click **Terraform Actions**.
4. Select **Destroy**.
5. A confirmation dialog will appear. Enter the stack name (`OCI_Benchmark_Stack`) to confirm.
6. Click **Destroy**.

This initiates a "Destroy" job. Resource Manager will systematically tear down all the resources (compute instance, block volume, VCN, subnets, etc.) that were created as part of this stack.

> 🧠 Important: Always use `terraform destroy` (or the Resource Manager equivalent) to clean up resources created by Terraform. Manually deleting resources in the OCI Console can lead to "drift" and make future Terraform operations unpredictable.

What to verify: Monitor the "Destroy" job logs. It typically takes a few minutes to complete. Once finished, the job status should be "Succeeded". You can also navigate to **Compute** > **Instances** and **Storage** > **Block Volumes** in the OCI Console to confirm that the resources have been terminated and deleted.

2. (Optional) Remove the Compartment and SSH Key

If you no longer plan to use the `BenchmarkAutomation` compartment or the uploaded SSH key, you can remove them.

1. **Delete SSH Key:**
    - Navigate to **Compute** > **SSH Keys**.
    - Find your `oci_benchmark_key` in the `BenchmarkAutomation` compartment.
    - Click the Actions menu (three dots) and select **Delete`.
2. **Delete Compartment:**
    - Navigate to **Identity & Security** > **Compartments**.
    - Find your `BenchmarkAutomation` compartment.
    - Click the Actions menu and select **Delete Compartment`.
    - You might need to confirm the deletion.

> ⚡ Note: You can only delete a compartment if it contains no resources. Ensure all resources (including any lingering network components or log buckets if your stack created them) are gone before attempting to delete the compartment.

Section Accomplishment

You have successfully cleaned up all OCI resources provisioned by the benchmark automation stack, preventing unwanted costs and maintaining a tidy OCI tenancy.

Common Issues and Troubleshooting

Even with automation, you might encounter issues. Here's a guide to some common problems and how to resolve them when working with OCI Resource Manager and Terraform.

1. Resource Manager Job Failures

- **Issue:** "Apply" or "Destroy" job fails with generic errors in Resource Manager.

- **Troubleshooting:**
    - **Review Job Logs:** This is your primary source of information. Resource Manager logs are very detailed. Look for specific error messages, resource names, or line numbers in the Terraform configuration.
    - **Permissions:** Often, the user or dynamic group used by Resource Manager lacks the necessary IAM policies to create, manage, or delete specific OCI resources. Verify your policies grant `manage` permissions for `compute-instances`, `block-volumes`, `vcns`, etc., within your target compartment.
    - **Resource Limits:** You might hit tenancy limits (e.g., maximum number of compute instances, VCNs, public IPs). Check **Governance & Administration** > **Tenancy Explorer** > **Service Limits**.
    - **Invalid Variables:** Double-check your stack variables (OCIDs, region, instance shape, image OCID). A typo in an OCID is a frequent culprit.
    - **Availability Domain Issues:** Ensure the `availability_domain` you specified is valid for your region and has capacity for the chosen instance shape.

2. SSH Connection Issues to Compute Instance

- **Issue:** You cannot SSH into the provisioned compute instance.
- **Troubleshooting:**
    - **Public IP:** Verify the instance has a public IP address.
    - **Security List/Network Security Group:** Ensure your VCN's security lists or Network Security Groups (NSGs) allow inbound SSH traffic (TCP port 22) from your client IP address.
    - **SSH Key:** Confirm you are using the correct private key (`~/.ssh/oci_benchmark_key`) and that its permissions are set correctly (`chmod 400 ~/.ssh/oci_benchmark_key`).
    - **User:** The default user for Oracle Linux images is `opc`. For Ubuntu, it's typically `ubuntu`. Ensure you're using the correct username.
    - **Public Key on Instance:** If you can't connect, it's possible the public key wasn't correctly injected. Check the `cloud-init-output.log` via the instance console connection or by attaching a temporary boot volume to another instance.

3. Benchmarks Not Running or Incomplete Results

- **Issue:** The `terraform apply` job succeeded, but benchmark logs are missing or incomplete when you SSH into the instance.
- **Troubleshooting:**
  - **`cloud-init` Logs:** The primary place to debug instance startup scripts is `/var/log/cloud-init-output.log` on the instance itself. SSH in and review this file for errors during package installation or script execution.
  - **Script Paths/Permissions:** Ensure the benchmark scripts within the Terraform stack have correct paths and executable permissions.
  - **Disk Space:** Verify the instance has enough disk space to install tools and store benchmark results.
  - **Dependencies:** Check if all necessary dependencies for `sysbench` and `fio` were successfully installed.

4. Terraform State Drift

- **Issue:** Resource Manager shows a "drift" warning, or subsequent `plan`/`apply` operations propose unexpected changes.
- **Troubleshooting:**
  - **Avoid Manual Changes:** The most common cause of drift is manually modifying OCI resources (e.g., changing an instance shape, deleting a block volume) outside of Resource Manager.
  - **Re-plan/Re-apply:** Sometimes, simply running a `plan` followed by an `apply` can resolve minor drift by bringing the infrastructure back into the desired state.
  - **Import Existing Resources:** For significant drift, if you want to bring manually created resources under Terraform management, you might need to use Terraform's `import` functionality (though this is more complex with Resource Manager and often better handled by recreating resources or manually adjusting state if you are an expert). For this tutorial, the best practice is not to manually modify resources.

> ⚠️ Common mistake: Manual modifications to OCI resources after Terraform deployment lead to configuration drift, making it difficult to track changes and potentially causing issues during subsequent Terraform operations. Always manage infrastructure changes through Terraform/Resource Manager.

Section Accomplishment

You are now equipped with knowledge to diagnose and troubleshoot common issues encountered during the deployment and execution of automated OCI benchmarks using Terraform and OCI Resource Manager.

What to Build Next

Congratulations on automating your OCI compute and storage benchmarks! This foundational setup opens doors to further enhancements and more sophisticated performance analysis. Here are three ideas to extend your project:

1. **Parameterize and Compare Different OCI Shapes/Tiers:** Modify the Terraform variables to easily switch between various OCI Compute instance shapes (e.g., E4.Flex vs. A1.Flex) or Block Volume performance tiers (Balanced, Higher Performance). Automate running the benchmarks for each configuration and storing results with unique identifiers, allowing for direct performance comparisons and cost-benefit analysis.
2. **Integrate with OCI Monitoring and Notifications:** Enhance the benchmark stack to push key performance metrics (IOPS, bandwidth, latency) to OCI Monitoring. Set up alarms that trigger OCI Notifications (e.g., email, PagerDuty) if benchmark results fall below expected thresholds. This creates a proactive system for detecting performance regressions or validating infrastructure changes.
3. **Automate Result Aggregation and Reporting:** Instead of manually SCPing logs, modify the benchmark scripts to upload results directly to an OCI Object Storage bucket. You could then use OCI Functions or another compute instance to process these raw logs, aggregate the data, and generate summary reports (e.g., CSV, HTML) for easier analysis and sharing. This moves towards a fully hands-off benchmarking pipeline.