

Lab 5.4

The Power of Tags — Beyond Compliance

Toepasbare Cloud Security voor IT-Professionals

Contents

1. The Power of Tags — Beyond Compliance	3
1.1. Context	3
1.2. Learning goals	3
1.3. Estimated time	3
1.4. Before you start	4
2. Explore Existing Tags on Your Resources	5
2.1. Find your resource group	5
2.2. Inspect tags on resources	5
2.3. Optional CLI check	5
3. Add Operational Tags	7
3.1. Add tags in the Azure portal	7
3.2. Optional CLI tagging	7
3.3. Question	8
4. Find Resources by Tag	9
4.1. Portal filtering	9
4.2. Azure Resource Graph	9
5. Tags for Cost Allocation	11
5.1. Open Cost Management	11
5.2. Budget discussion	12
6. Simulate Incident Response Using Tags	13
6.1. Scenario	13
6.2. Find the tagged resource	13
6.3. Discuss possible response actions	13
6.4. Resolve the incident	14
7. Tags for Lifecycle Management	15
7.1. Scenario: decommissioning	15
7.2. Tag a test resource	15
8. Tags and Automation	16
8.1. Automation design exercise	16
9. Reflection Questions	17
10. Summary	18

1. The Power of Tags — Beyond Compliance

1.1. Context

In Labs 5.2 and 5.3, you used tags as part of compliance and governance. You saw that Azure Policy can require tags or inherit tags from a resource group.

But tags are more than a compliance checkbox.

Tags connect cloud resources to the operational processes around them:

- Who owns this resource?
- Which environment does it belong to?
- Which cost centre should pay for it?
- Is it part of an incident investigation?
- Is it active, temporary, or ready for decommissioning?
- Which resources belong to the same application, system, or course day?

In this lab, you will explore tags as a practical security and operations tool.

Note

This lab is exploratory. The goal is not to memorize every Azure tagging rule, but to understand how tags make cloud environments easier to manage, investigate, secure, and automate.

Note

Tags are stored as plain text. Do not put secrets, passwords, access tokens, personal data, or sensitive incident details in tag names or tag values.

1.2. Learning goals

By the end of this lab, you should be able to explain:

- Why tags are useful beyond compliance.
- How tags support ownership, cost allocation, incident response, and lifecycle management.
- How to inspect and update tags in Azure.
- Why tag consistency matters at scale.
- What the limitations of tags are.
- How tags can be used by automation, without assuming that tags trigger automation by themselves.

1.3. Estimated time

45 minutes

Suggested timing:

Time	Activity
5 minutes	Explore existing tags on Day 5 resources
10 minutes	Add operational tags

10 minutes	Find and query resources by tag
10 minutes	Review cost and ownership use cases
5 minutes	Simulate incident and lifecycle use cases
5 minutes	Reflection and discussion

1.4. Before you start

Make sure you have:

- ☐ Day 5 infrastructure deployed.
- ☐ Lab 5.2 and Lab 5.3 completed or reviewed.
- ☐ Contributor access on your lab resource group.
- ☐ Access to the Azure portal.
- ☐ Azure CLI available if you want to use the optional CLI commands.

Note

Azure allows up to 50 tags on a resource, resource group, or subscription. Not every resource type supports tags, and not every service passes tags into cost data in the same way.

2. Explore Existing Tags on Your Resources

Your Day 5 infrastructure should already have tags applied by Terraform.

Typical tags may include:

```
environment = dev
student     = <your_prefix>
managed_by  = terraform
day         = day-5
```

These tags help identify what the resource belongs to and how it was created.

2.1. Find your resource group

1. In the Azure portal, search for **Resource groups**.
2. Open your Day 5 resource group.

Example:

```
rg-<your_prefix>-d5
```

3. Open the **Tags** blade on the resource group.
4. Write down the tags you see.

Tag name	Tag value	What does this tell us?

2.2. Inspect tags on resources

Open several resources in the resource group and check their tags.

Look at resources such as:

- Public backend App Service.
- Internal backend App Service.
- Key Vault.
- Storage account.
- Log Analytics workspace.
- Application Insights component.

Question

Which resources have tags? Which resources are missing tags? Why might some Azure resources or child resources not show the same tags?

2.3. Optional CLI check

Use Azure CLI to list resources and their tags:

```
az resource list `
  --resource-group <RESOURCE_GROUP_NAME> `
  --query "[].{name:name,type:type,tags:tags}" `
  --output table
```

Note

The examples use PowerShell-style line continuation with a backtick. If you use Bash, replace the backtick with a backslash.

3. Add Operational Tags

Now add tags that reflect real-world operational needs.

For this lab, use the following tag set:

Tag	Example value	Purpose
cost-centre	IT-12345	Cost allocation
owner	<your-name-or-team>	Accountability
status	active	Lifecycle tracking
application	cello-demo	Application grouping

Note

In a real organization, tag names and allowed values should be standardized. For example, use either `cost-centre` or `costCenter`, but do not allow every team to invent its own spelling.

3.1. Add tags in the Azure portal

1. Open your resource group.
2. Select a resource.
3. Open **Tags**.
4. Add the operational tags from the table.
5. Save the changes.

Repeat this for a few important resources.

Recommended resources:

- Public backend App Service.
- Internal backend App Service.
- Key Vault.
- Storage account.

3.2. Optional CLI tagging

First get the resource ID:

```
az resource show `
  --resource-group <RESOURCE_GROUP_NAME> `
  --name <RESOURCE_NAME> `
  --resource-type <RESOURCE_TYPE> `
  --query id `
  --output tsv
```

Then merge the new tags into the existing tag set:

```
az tag update `
  --resource-id <RESOURCE_ID> `
  --operation Merge `
  --tags cost-centre=IT-12345 owner=<your-name-or-team> status=active
application=cello-demo
```

Note

Use merge instead of replace when you want to keep existing tags. Replacing tags can accidentally remove tags created by Terraform, Azure Policy, or another team.

3.3. Question

Question

Azure allows a maximum of 50 tags per resource. Why is that limit important when designing a tagging strategy?

4. Find Resources by Tag

Tags become useful when you can search, group, and filter by them.

4.1. Portal filtering

1. Open your resource group.
2. Use the tag filter in the resource list.
3. Filter for:

```
application = cello-demo
```

4. Then filter for:

```
status = active
```

Question

How would this help an operations team during troubleshooting?

4.2. Azure Resource Graph

Azure Resource Graph is useful when you want to query resources across a subscription or larger scope.

Open **Resource Graph Explorer** in the Azure portal and run:

```
Resources
| where resourceGroup =~ "<RESOURCE_GROUP_NAME>"
| project name, type, resourceGroup, tags
| order by name asc
```

Find resources with a specific tag:

```
Resources
| where resourceGroup =~ "<RESOURCE_GROUP_NAME>"
| where tostring(tags["application"]) == "cello-demo"
| project name, type, resourceGroup, application=tostring(tags["application"]),
owner=tostring(tags["owner"]), status=tostring(tags["status"])
| order by name asc
```

Find resources missing an owner tag:

```
Resources
| where resourceGroup =~ "<RESOURCE_GROUP_NAME>"
| where isempty(tostring(tags["owner"]))
| project name, type, resourceGroup, tags
| order by name asc
```

Note

Azure Resource Graph uses a KQL-like query language over Azure Resource Manager resource metadata. It is useful for inventory, governance, and large-scale resource discovery.

Question

Which is more useful at scale: clicking through each resource manually, or querying missing tags with Resource Graph? Why?

5. Tags for Cost Allocation

Tags help turn raw cloud spend into something a business can understand.

Instead of only seeing:

Total subscription cost

you can group costs by:

- Application.
- Environment.
- Cost centre.
- Team or owner.
- Course day or project.

5.1. Open Cost Management

1. In the Azure portal, search for **Cost Management + Billing**.
2. Open **Cost analysis**.
3. Set the scope to your subscription or resource group, depending on what you can access.
4. Set the date range to the current month.
5. Group by a tag such as:

cost-centre

Then try grouping by:

day

or:

application

Note

Cost data can take time to appear. Tag-based cost reporting usually applies to usage after the tag was applied. If you do not see cost data immediately, verify the tags on the resources and discuss the expected cost view instead.

Note

Some costs may appear as untagged. Common reasons are: the resource was not tagged when the usage was emitted, the resource type does not emit tags into cost data, or the cost belongs to a related service or child resource.

Question

Why is cost grouped by `application` or `cost-centre` more useful than only looking at total subscription spend?

5.2. Budget discussion

You do not need to create a budget in this lab, but discuss how a budget could use tags.

Example:

Alert when resources tagged cost-centre = IT-12345 exceed their expected monthly spend.

Question

Which tags would finance need? Which tags would the application team need? Which tags would security need?

6. Simulate Incident Response Using Tags

Tags can help coordinate incident response, but they are not a security boundary.

A tag does not isolate a resource. A tag does not block traffic. A tag does not revoke access.

A tag helps people and automation find the right resources quickly.

6.1. Scenario

You detect unusual behavior on the public backend App Service.

You want to mark it as part of an active investigation.

Add these tags to the public backend App Service:

Tag	Value
incident-status	investigating
incident-priority	medium

Note

Do not put sensitive investigation details in tags. Tags are metadata and may be visible to many people who can read the resource.

6.2. Find the tagged resource

Use the portal tag filter or Resource Graph:

```
Resources
| where resourceGroup =~ "<RESOURCE_GROUP_NAME>"
| where tostring(tags["incident-status"]) == "investigating"
| project name, type, resourceGroup, incidentStatus=tostring(tags["incident-status"]), priority=tostring(tags["incident-priority"])
```

Question

How would tags help if the incident affected 30 resources across multiple resource groups?

6.3. Discuss possible response actions

A real response could include:

Review Application Insights logs. Review Activity Log events. **Check recent deployments or configuration changes.** Add or tighten App Service access restrictions. **Disable a compromised credential or identity.** Notify the owner listed in the owner tag.

Note

Only take containment actions if your instructor asks you to. In this lab, the tag is the main exercise. The containment actions are discussion points.

6.4. Resolve the incident

After the simulated investigation, either remove the incident tags or change the status:

`incident-status = resolved`

Question

Why is it important to update or remove incident tags after the investigation?

7. Tags for Lifecycle Management

Tags can also support lifecycle automation.

7.1. Scenario: decommissioning

Imagine your organization uses the following lifecycle tag:

```
status = decommissioned
```

A scheduled automation checks for this tag and starts a controlled removal process.

The process might be:

Find resources with status = decommissioned. Notify the owner. **Wait for an approval period.** Export or back up required data. **Delete the resource.** Record the action in a ticket or audit log.

Note

Changing a tag should not immediately delete resources unless the organization has strong safeguards. A delay, approval, backup, and ownership check reduce the chance of accidental data loss.

7.2. Tag a test resource

Pick a non-critical lab resource and change:

```
status = active
```

to:

```
status = decommissioned
```

Then find it with Resource Graph:

Resources

```
| where resourceGroup =~ "<RESOURCE_GROUP_NAME>"  
| where tostring(tags["status"]) == "decommissioned"  
| project name, type, resourceGroup, owner=tostring(tags["owner"]),  
status=tostring(tags["status"])
```

Question

Why is a 30-day delay safer than deleting a resource immediately when its status tag changes?

8. Tags and Automation

Tags are often used as input for automation.

Common automation patterns:

- A scheduled job queries Azure Resource Graph for resources with a certain tag.
- Azure Policy adds or modifies tags.
- A Logic App or Azure Automation runbook acts on resources with a specific tag.
- Cost Management uses tags for reporting and budgets.
- Security tooling uses tags to prioritize investigation.

Note

Tags are metadata. They do not automatically run code. You need a separate mechanism, such as Azure Policy, Azure Resource Graph plus a scheduled job, Event Grid, Logic Apps, Azure Automation, or a CI/CD pipeline.

8.1. Automation design exercise

Design a simple automation around this tag:

`status = decommissioned`

Fill in the table:

Question	Your design
How does the automation find the resource?	
How often does it run?	
Who gets notified?	
What approval is needed?	
What backup is required?	
What happens after the waiting period?	
How is the action logged?	

9. Reflection Questions

Your answer:

1. Which tag use case was most useful: cost allocation, ownership, incident response, or lifecycle management? Why?
2. If you designed a tagging strategy for an organization, which tags would be required on every resource?
3. Which tags should be optional?
4. Which tag values should be restricted to an approved list?
5. Why should secrets, personal data, or detailed incident information not be stored in tags?
6. How would you enforce consistent tags at scale?

10. Summary

In this lab, you explored tags beyond compliance.

You used tags for:

- Ownership and accountability.
- Cost allocation and reporting.
- Resource discovery.
- Incident coordination.
- Lifecycle management.
- Automation design.

The key lesson is:

“Tags are not security controls by themselves, but they make security, operations, cost management, and automation easier to execute at scale.”

Tags are most valuable when they are consistent, meaningful, and connected to real processes.