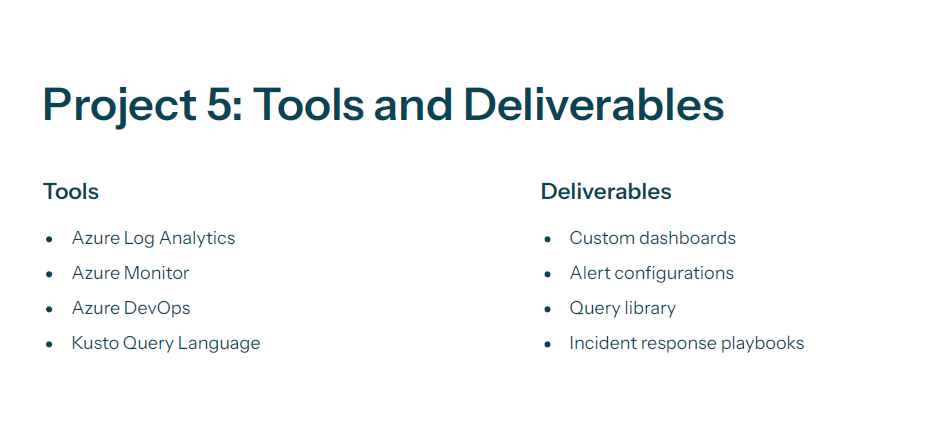
**Project-5 Azure Monitoring → Azure DevOps Bug Integration**

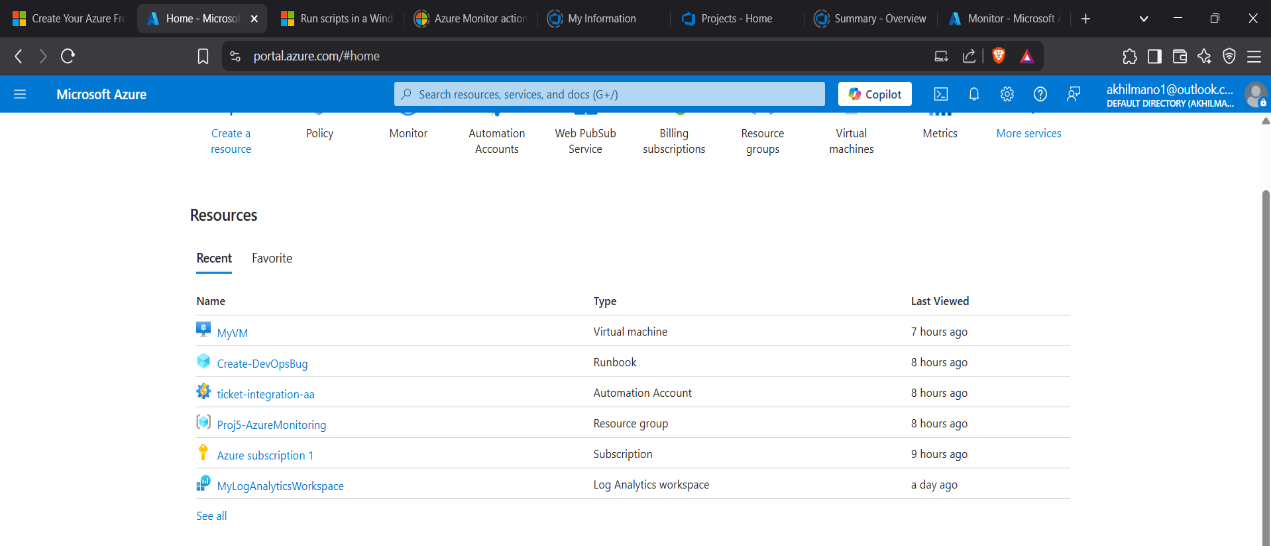
This document captures every step taken to implement an end‑to‑end flow that:

1. Monitors VM CPU via Azure Monitor & Log Analytics
2. Fires an alert when CPU > 80%
3. Triggers an Azure Automation runbook
4. Creates a Bug work item in Azure DevOps
5. **Prerequisites**

****

* **Azure subscription** with contributor access
* **Azure DevOps** organization (Nithin) and project (AZurearm)
* **Visual Studio Code** or PowerShell for scripting
* **Remote access** to the target VM (Windows)

1. **Prepare the VM and Log Analytics Workspace**

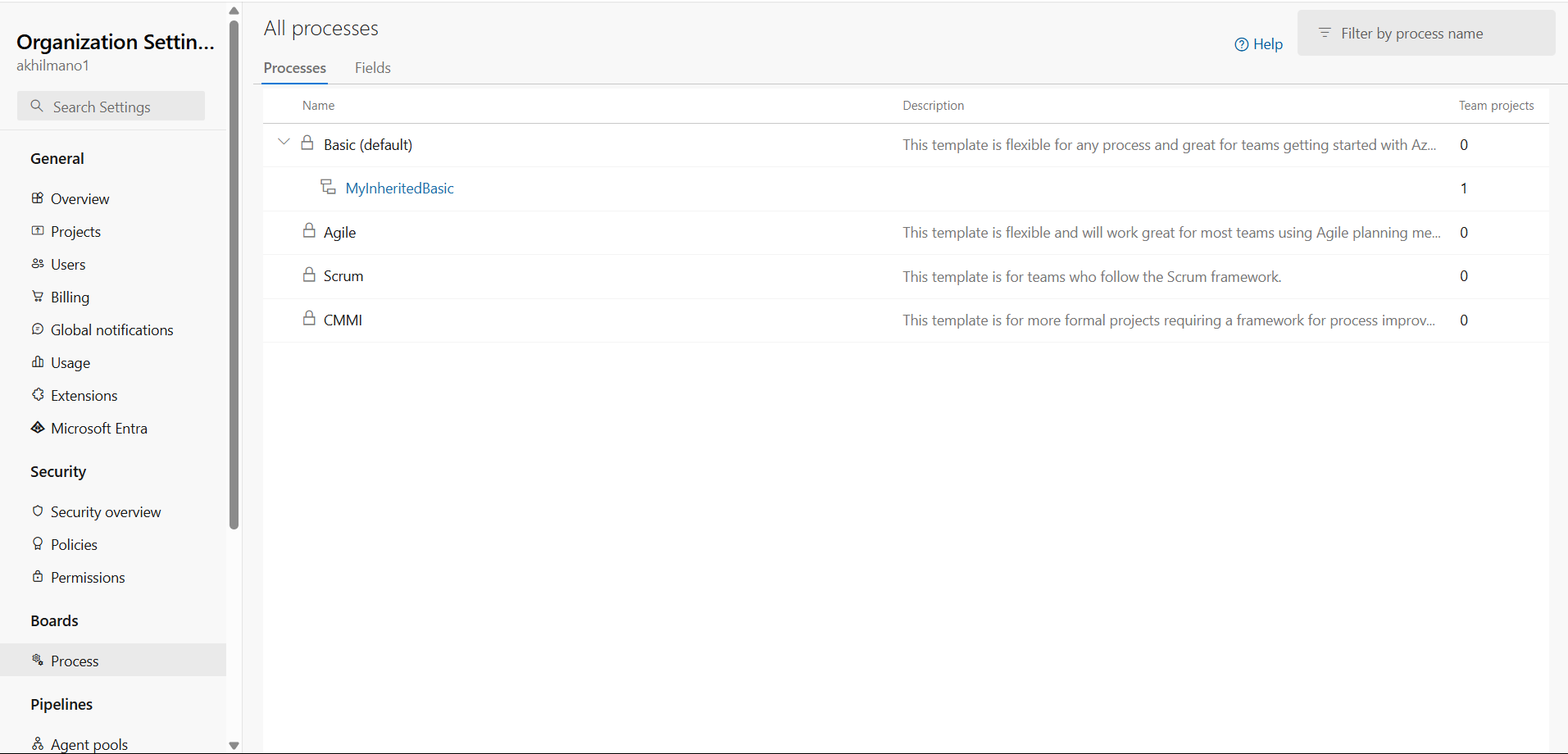
****

1. **Create Log Analytics workspace**
   * In Azure portal: **Monitor → Logs → Create**
   * Name: MyLogAnalyticsWorkspace (in resource group Proj5-AzureMonitoring)
2. **Install & connect VM agent**
   * On VM resource: **Connect → Agents → Link workspace**
   * Select MyLogAnalyticsWorkspace
3. **Verify heartbeat data**
   * In workspace: **Logs → Run Heartbeat | take 10**
   * Observe recent TimeGenerated entries

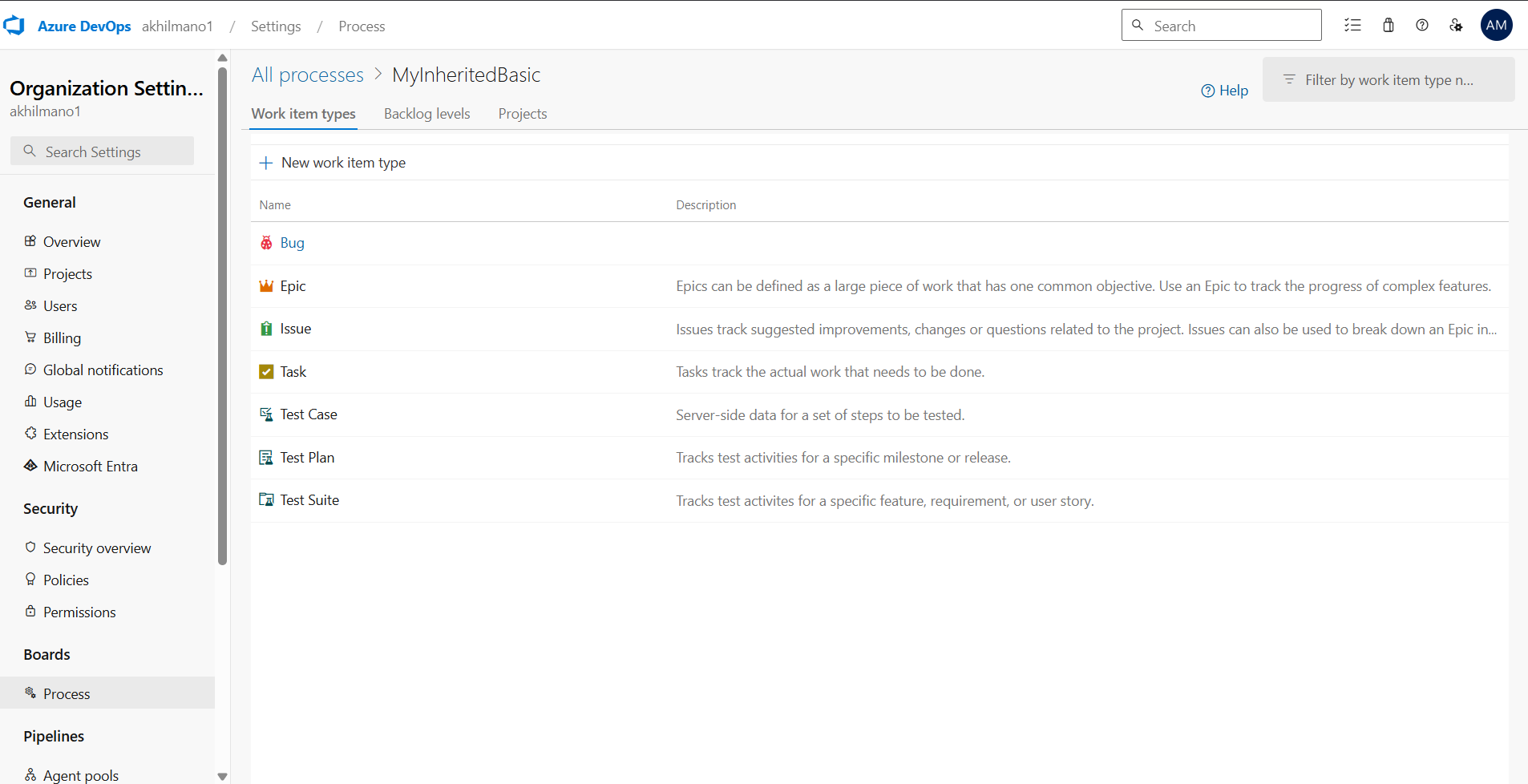
**3. Set up Azure DevOps Work Item Type**

Azure DevOps Basic process does not include Bug by default, so we must:

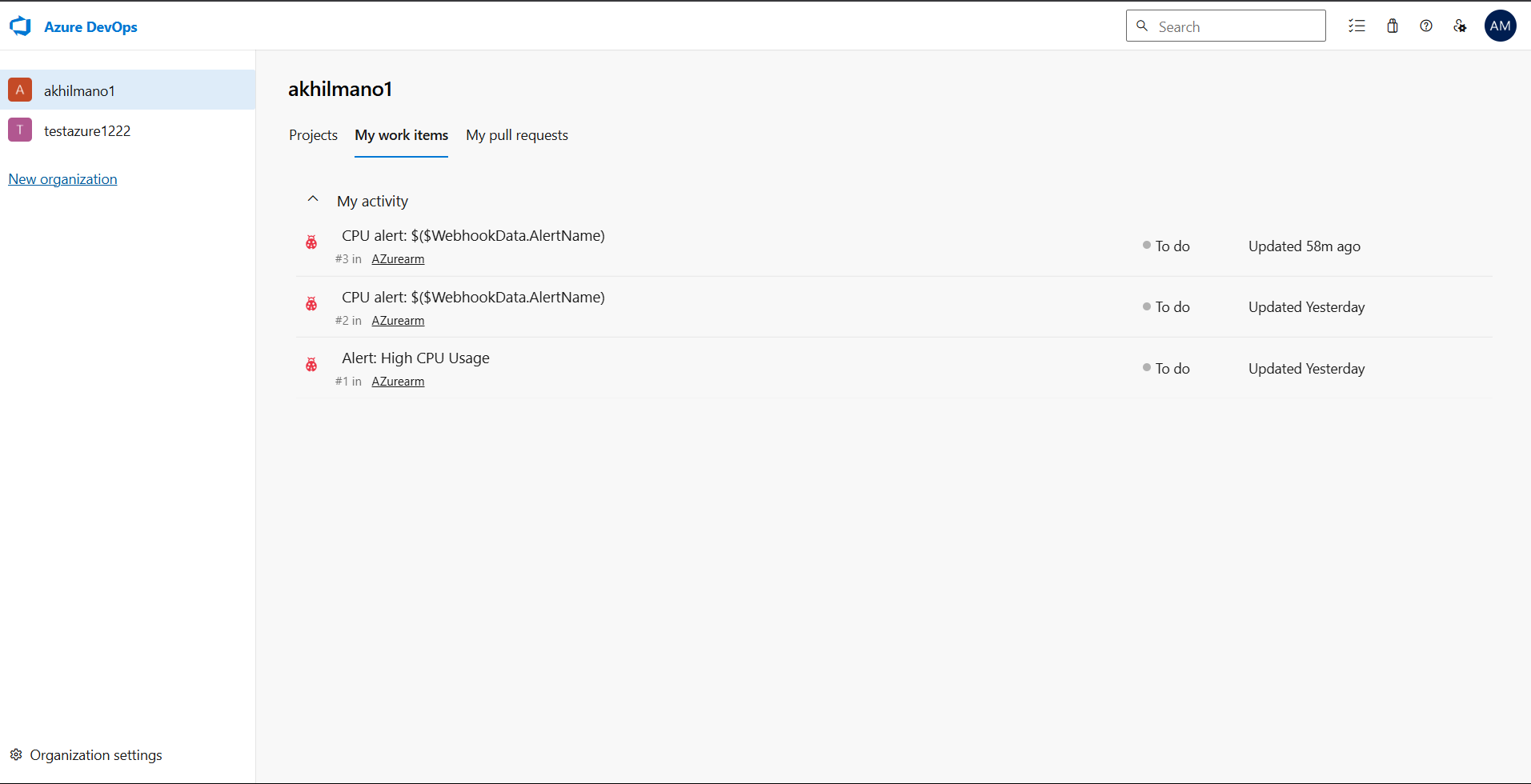
1. **Create inherited process**
   * Org level: **Organization settings (⚙️) → Boards → Process → New inherited process**
   * Base: **Basic**, Name: MyInheritedBasic



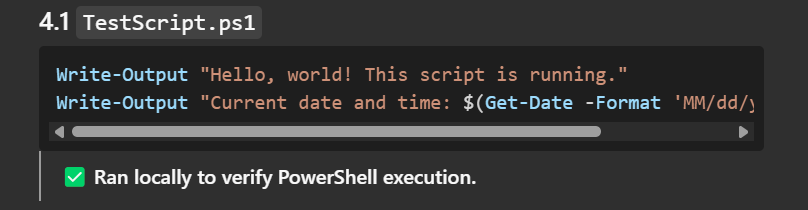
1. **Add Bug work item type**
   * Process: **MyInheritedBasic → Work item types → + New work item type**
   * Choose built‑in **Bug**

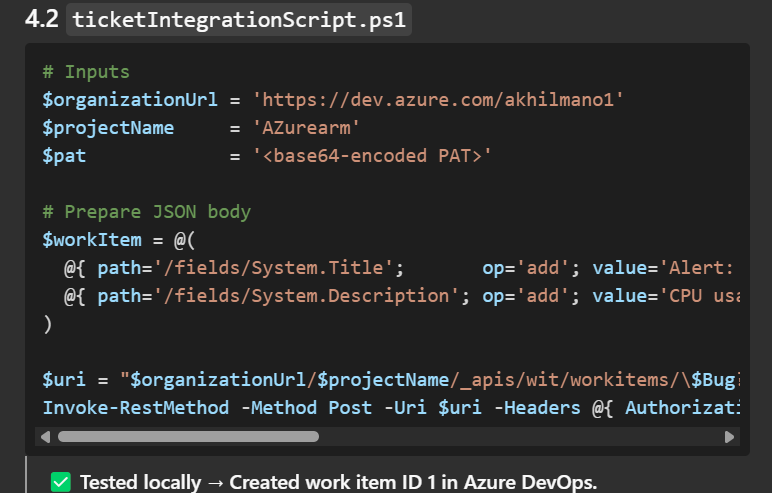


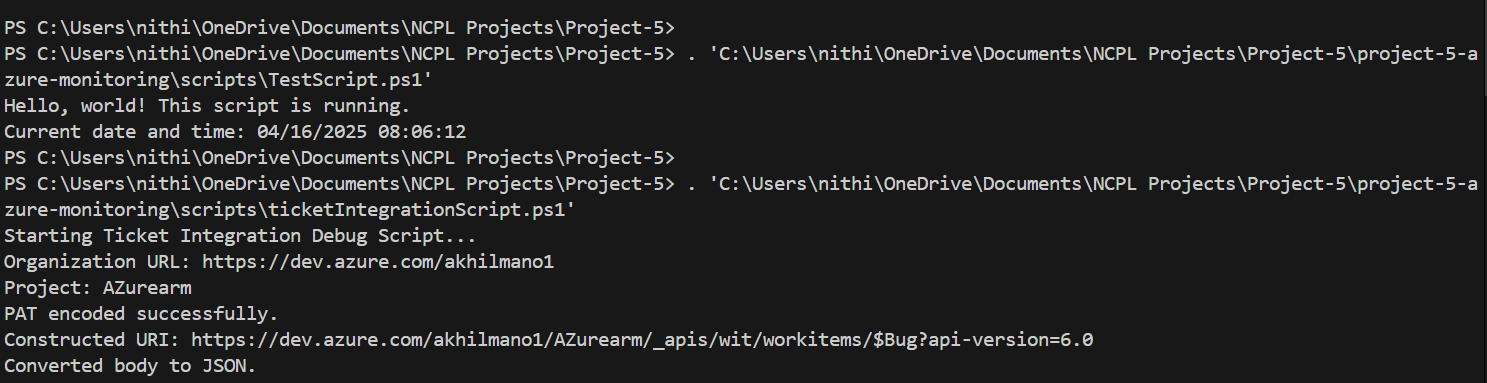
1. **Associate project with new process**
   * Org settings → **Projects** → AZurearm → **Change process** → MyInheritedBasic
2. **Verify**
   * Project: **Boards → Work items** → Filter **Type = Bug**



**4. Local Script Testing**



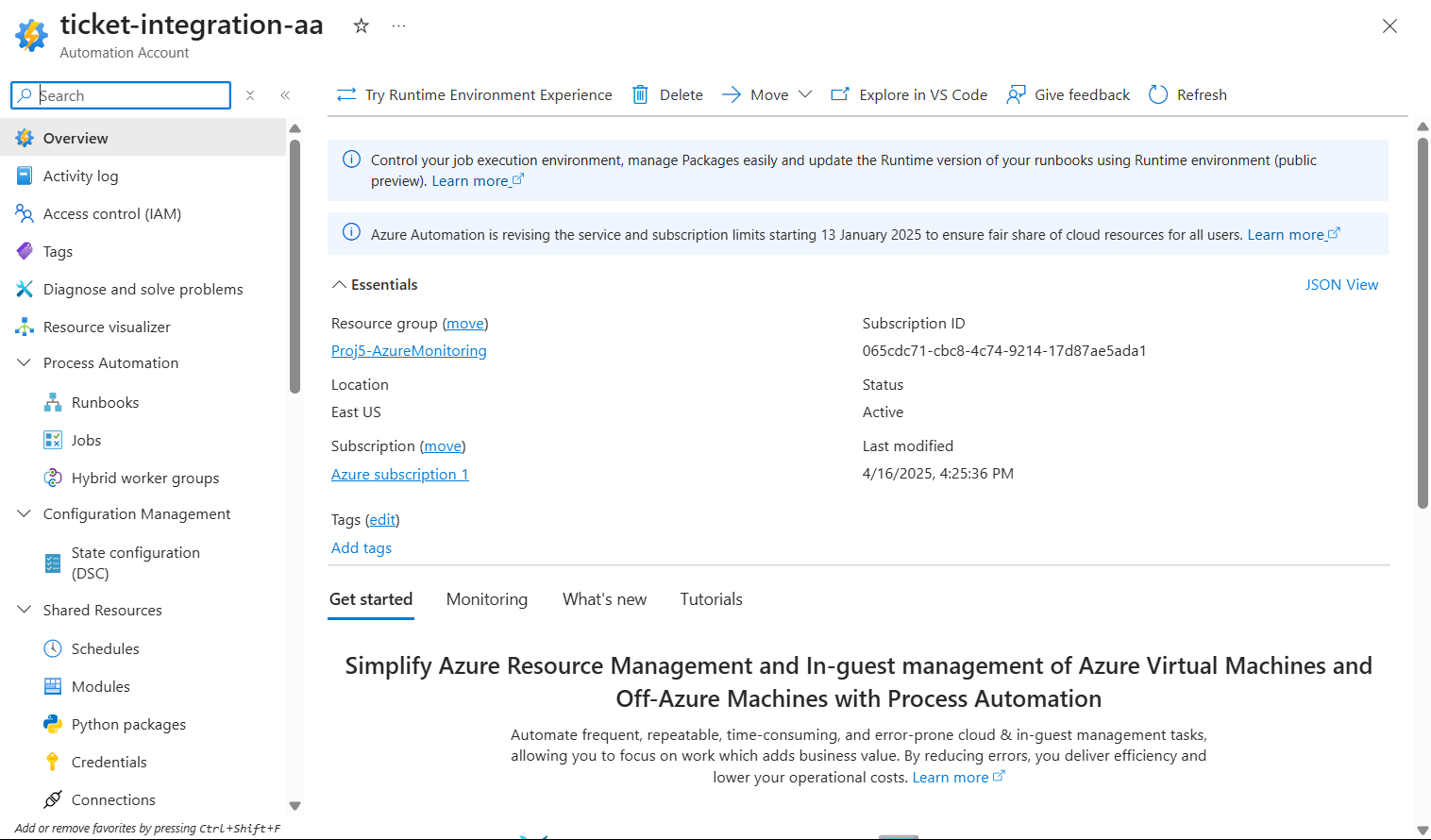




**5. Configure Azure Automation**

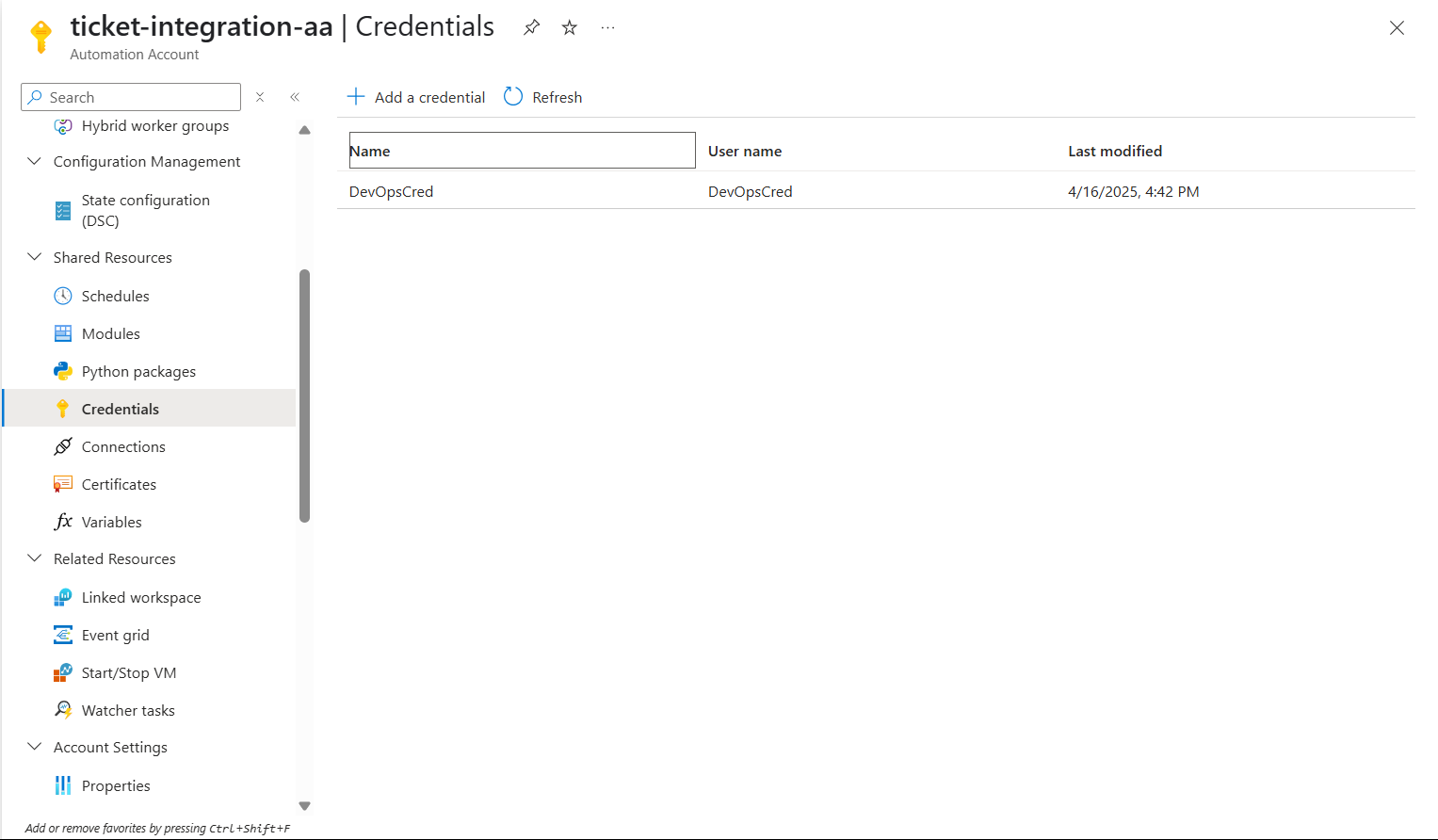
**5.1 Create Automation Account**

* Name: ticket-integration-aa
* Resource Group: Proj5-AzureMonitoring



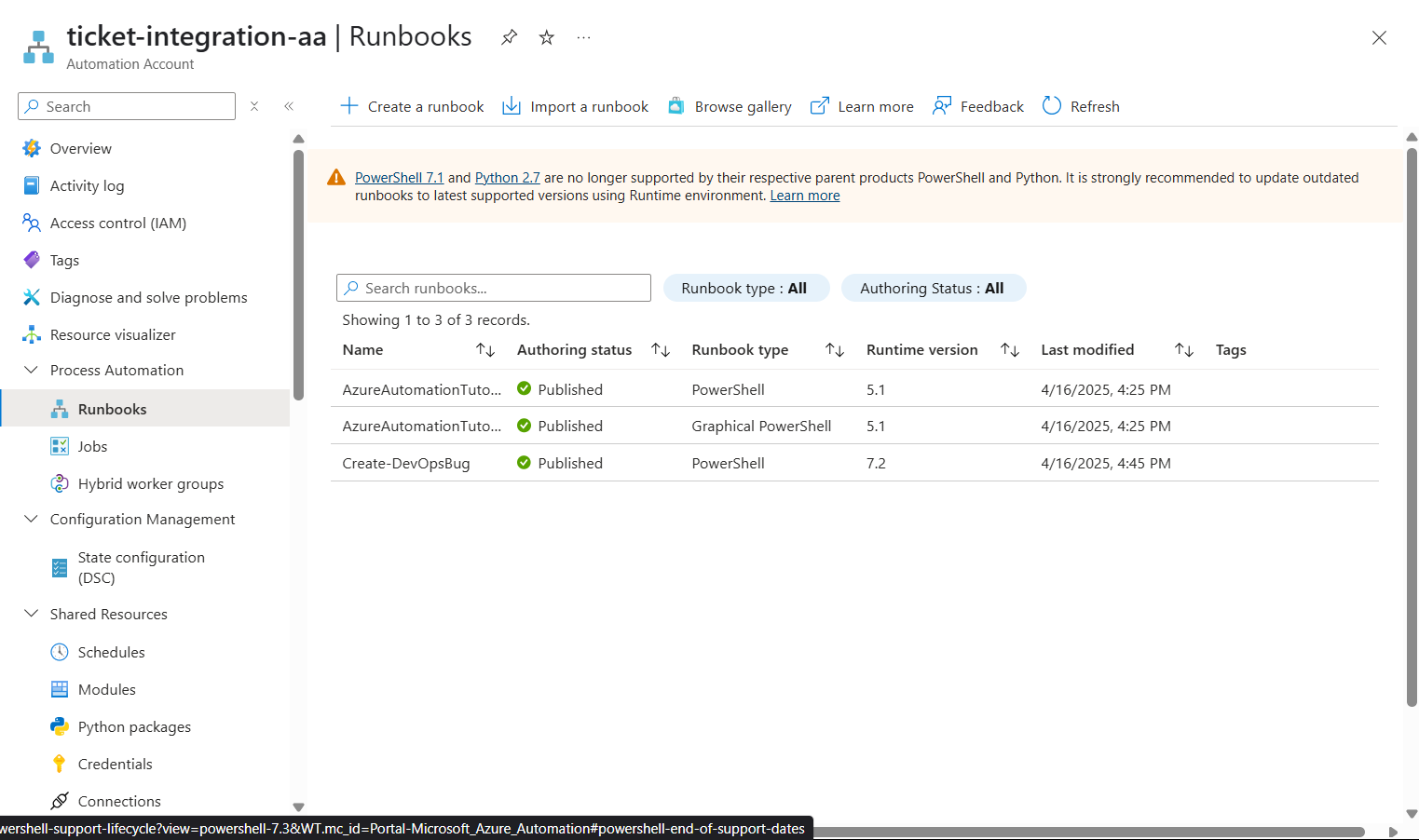
**5.2 Store PAT Securely**

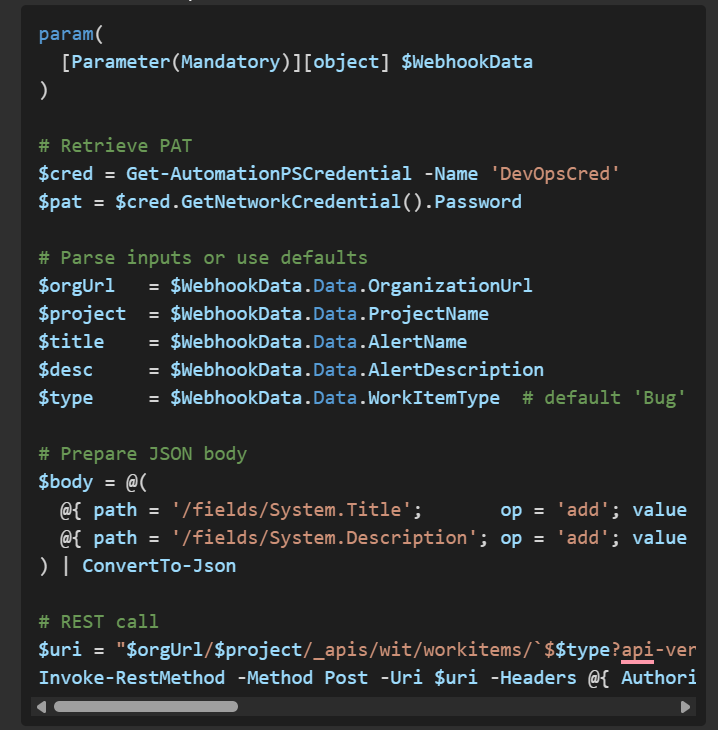
* In **Shared Resources → Credentials**:
  + **Name**: DevOpsCred
  + **Username**: (dummy)
  + **Password**: Personal Access Token



**5.3 Create and Publish Runbook**

1. **Process Automation → Runbooks → + Create a runbook**
   * Name: Create-DevOpsBug
   * Type: **PowerShell**, Version: **7.2**

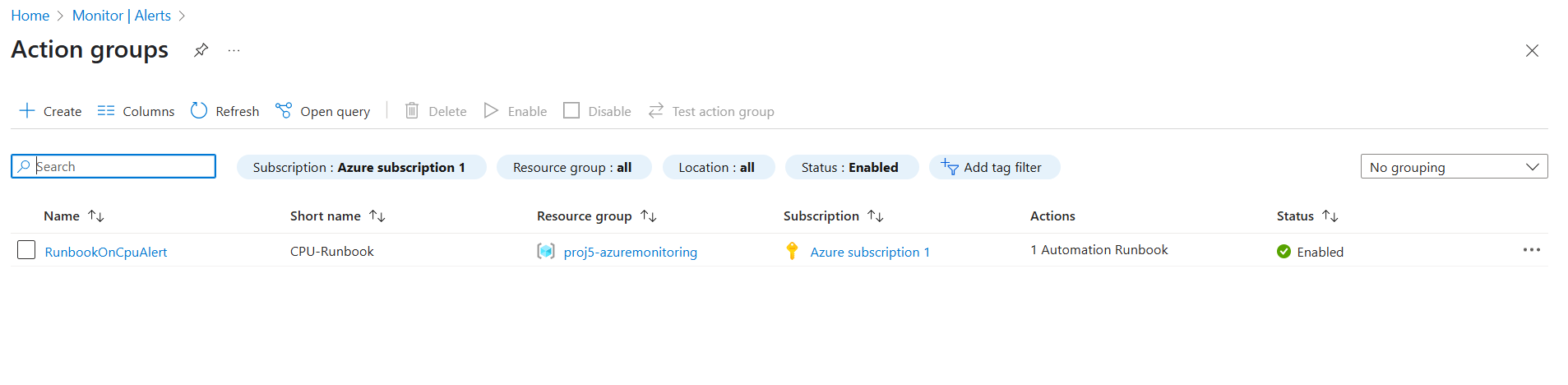




Click **Save** → **Publish**

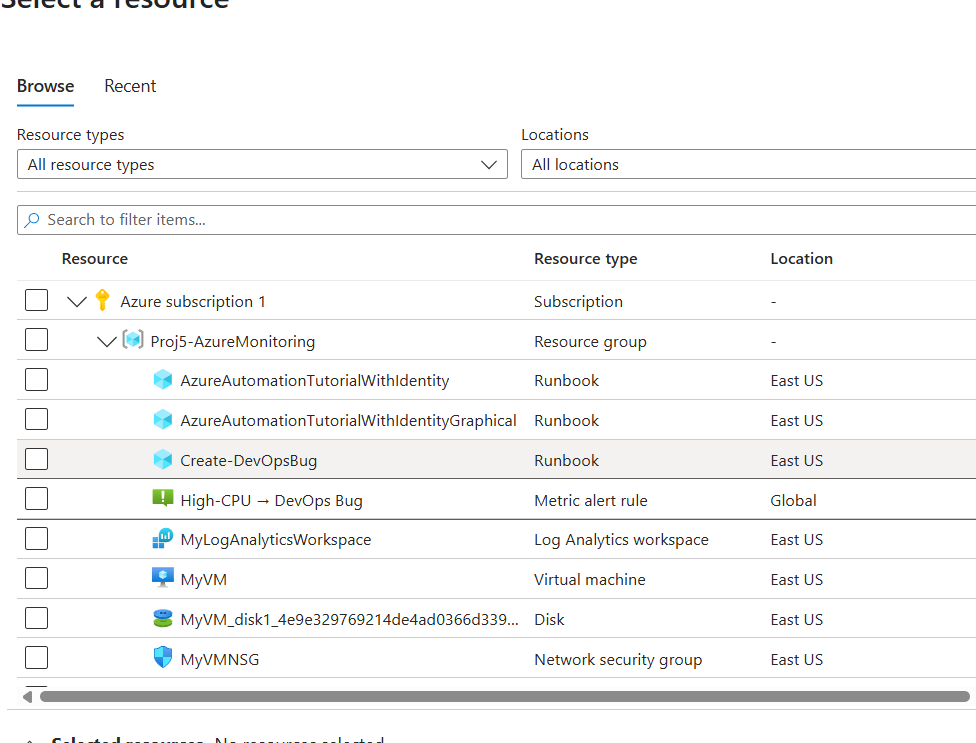
**6. Create Action Group**

1. **Monitor → Alerts → Action groups → + Create**
2. Basics: Resource group Proj5-AzureMonitoring, Name/Display name
3. **Actions → Action type**: **Automation Runbook**
   * Subscription: Azure subscription 1
   * Automation account: ticket-integration-aa
   * Runbook: Create-DevOpsBug
   * Enable common alert schema: **No**
4. **Tags**, then **Review + create** → **Create**



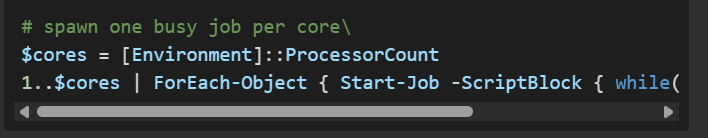
**7. Create Metric Alert Rule**

1. **Monitor → Alerts → + Create → Metric alert rule**
2. **Scope**: MyVM
3. **Condition**:
   * Signal: **Percentage CPU** (Maximum)
   * Operator: **Greater than** 80
   * Aggregation: **Maximum** over **5 min**, evaluated every **1 min**
4. **Actions**: select the action group created above
5. **Details**: give the rule a name (e.g. HighCpuAlert)
6. **Review + create** → **Create**

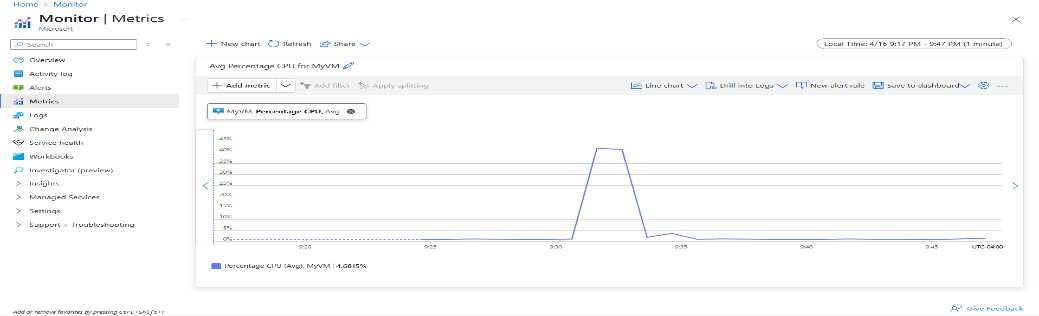


**8. Test End‑to‑end Flow**

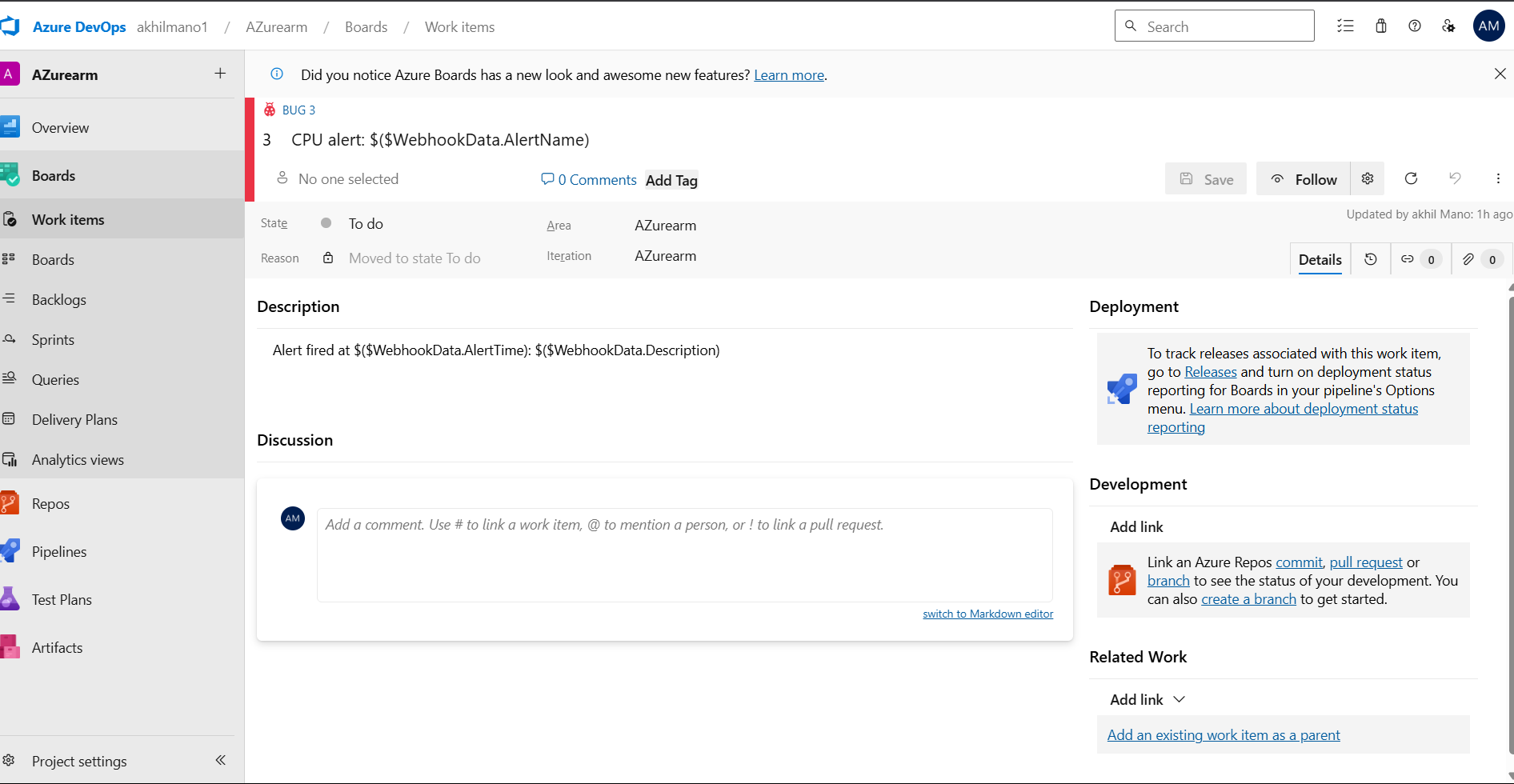
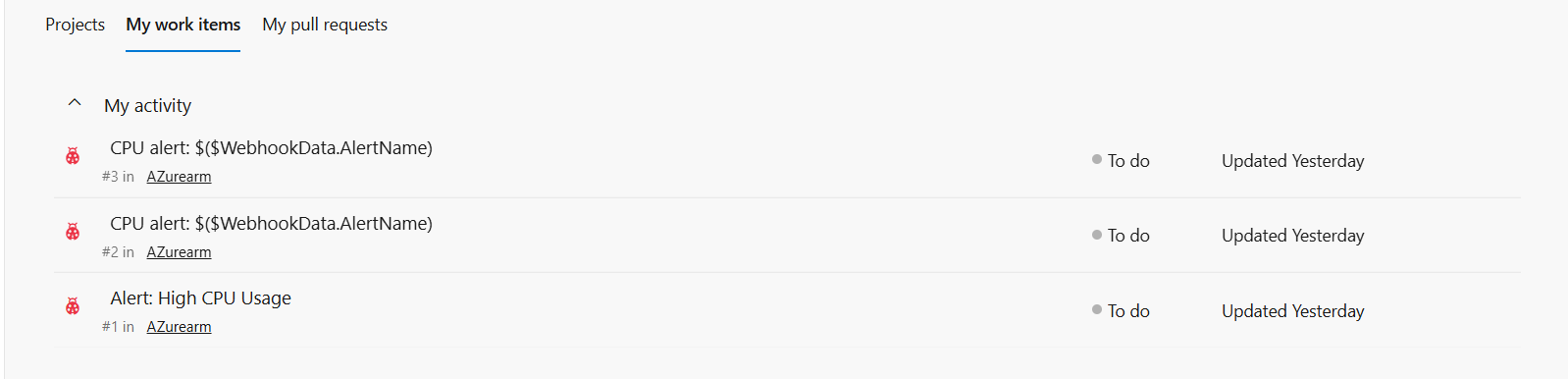
1. **Generate CPU spike** on MyVM (via Azure Portal Run Command):



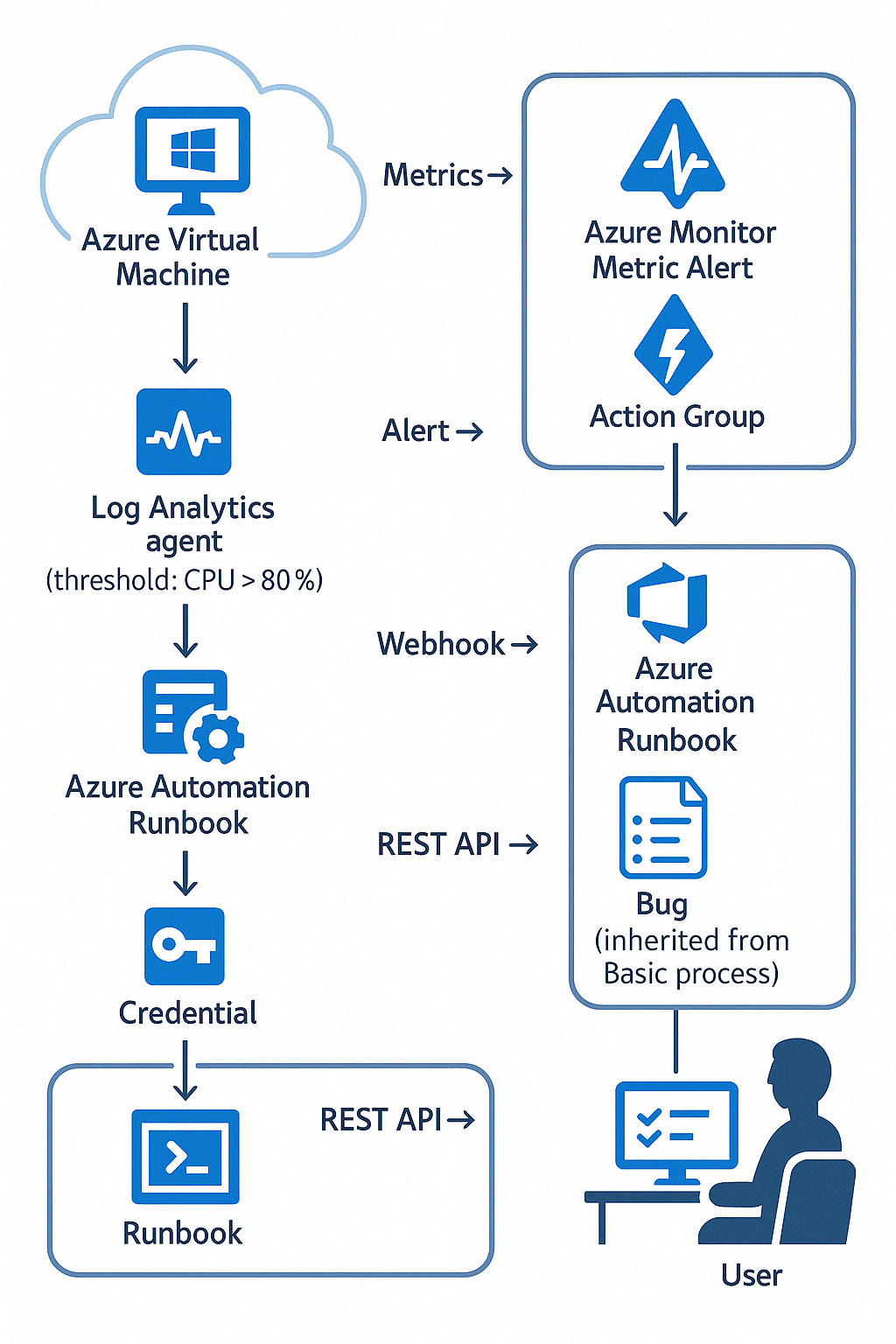
1. **Wait ~5 minutes** for the metric alert to trigger → action group → runbook



1. **Verify runbook job**
   * Azure portal: **Automation Accounts → ticket-integration-aa → Jobs**
   * Look for Create-DevOpsBug entry, ensure Status = Completed
2. **Verify work item**
   * Azure DevOps: **Boards → Work Items → filter Type = Bug**
   * Confirm title & description



High Level Architecture Diagram



High Level Architecture Diagram