Microsoft Fabric Workspace Provisioning Guide

A Complete Step-by-Step Tutorial for Running Fabric Workspace Scenarios

---

Document Information: - **Version:** 1.0 - **Date:** October 2024 - **Project:** USF Fabric CI/CD - **Repository:** usf-fabric-cicd

---

Table of Contents

1. Prerequisites
2. Environment Setup
3. Scenario 1: Config-Driven Workspace
4. Scenario 2: Domain-Workspace
5. Scenario 3: Feature Branch Workflow
6. Comparison Matrix
7. Troubleshooting
8. Next Steps

Prerequisites

Required Software

Software	Minimum Version	Check Command
Conda/Miniconda	Latest	`conda --version`
Python	3.9+	`python --version`
Git	2.x	`git --version`

Required Azure Resources

- Azure Service Principal with Fabric permissions
- Microsoft Fabric workspace capacity (for lakehouse/warehouse creation)
- Azure Active Directory access for user management

Required Files

```
usf-fabric-cicd/
  .env                    # Azure credentials (REQUIRED)
  project.config.json     # Naming patterns (for config-driven)
  requirements.txt        # Python dependencies
  scenarios/              # Scenario scripts
```

Environment Setup

Step 1: Navigate to Project Directory

```
cd /home/sanmi/Documents/J\'TOYE_DIGITAL/LEIT_TEKSYSTEMS/1_Project_Rhico/usf-fabric-cicd
```

Step 2: Verify Conda Environment

```
# List all conda environments
conda info --envs

# Look for: fabric-cicd (should have * if active)
```

If not active:

```
conda activate fabric-cicd
```

If environment doesn't exist:

```
conda create -n fabric-cicd python=3.9 -y
conda activate fabric-cicd
pip install -r requirements.txt
```

Step 3: Verify Python Version

```
python --version
# Expected: Python 3.9.x or higher
```

Step 4: Check Required Dependencies

```
pip list | grep -E "msal|requests|pyyaml"
```

Expected Output:

```
msal                4.x.x
PyYAML              6.x.x
requests            2.x.x
```

If missing:

```
pip install -r requirements.txt
```

Step 5: Verify Environment Variables

```
# Check .env file exists and has required variables
cat .env | grep -E "AZURE_CLIENT_ID|AZURE_TENANT_ID|FABRIC_CAPACITY_ID"
```

Expected Output:

```
AZURE_CLIENT_ID=your-service-principal-id
AZURE_TENANT_ID=your-tenant-id
AZURE_CLIENT_SECRET=your-secret
FABRIC_CAPACITY_ID=your-capacity-guid
```

If variables are missing, edit .env:

```
nano .env
```

Step 6: Run Preflight Check

```
./setup/preflight_check.sh
```

This verifies: - Conda environment active - Python version 3.9+ - Required packages installed - .env file exists and valid - Azure credentials configured - Git repository status

---

Scenario 1: Config-Driven Workspace

Best for: Enterprise environments with standardized naming conventions

Overview

This scenario uses `project.config.json` to generate workspace names following organizational patterns:

```
Input: --project analytics --environment dev
Output: usf2-fabric-analytics-dev (from pattern: {prefix}-{project}-{environment})
```

Prerequisites

Verify `project.config.json` exists:

```
ls -la project.config.json
```

If missing, initialize it:

```
python setup/init_project_config.py
```

Basic Usage (Trial Workspace)

Creates a Trial workspace without lakehouses/warehouses:

```
python scenarios/config-driven-workspace/config_driven_workspace.py \
  --project analytics \
  --environment dev
```

What gets created: - Workspace: `usf2-fabric-analytics-dev` - Principals template: `config/principals/analytics_dev_principals.txt` - Setup

log: `config/setup-logs/analytics_dev_setup_log.json` - No lakehouses (Trial limitation)

Console Output:

```
======================================================================
  Config-Driven Workspace Provisioning
======================================================================

  Configuration:
    Project Name:    analytics
    Environment:     dev
    Workspace Name:  usf2-fabric-analytics-dev (generated from config)
    Config Prefix:   usf2-fabric
    Naming Pattern:  {prefix}-{name}-{environment}

======================================================================

======================================================================
  STEP 1: Creating Workspace (Config-Driven)
======================================================================

  Creating workspace 'usf2-fabric-analytics-dev'...
    Description: Analytics workspace - Development environment
    Auto-deploy: True
    Requires approval: False
       No capacity ID - using Trial (lakehouse creation will fail)

  Workspace created successfully
  Workspace ID: abc123-def456-...
  Display Name: usf2-fabric-analytics-dev
  Type: Workspace
```

Full Usage (With Capacity)

Creates workspace with lakehouses and warehouses:

```
python scenarios/config-driven-workspace/config_driven_workspace.py \
  --project analytics \
  --environment dev \
  --capacity-id $FABRIC_CAPACITY_ID
```

What gets created: - Workspace: `usf2-fabric-analytics-dev` - Lakehouse: `USF2_FABRIC_Lakehouse_Dev` - Principals template - Setup log

Console Output:

```
======================================================================
  STEP 2: Creating Fabric Items
```

```
======================================================================

  Creating lakehouse: USF2_FABRIC_Lakehouse_Dev
  Created lakehouse: USF2_FABRIC_Lakehouse_Dev (ID: xyz789-abc123-...)
```

Adding Users (Interactive)

After workspace creation, you'll see:

```
======================================================================
  STEP 3: Configuring Workspace Principals
======================================================================

  Creating principals template: config/principals/analytics_dev_principals.txt
    Template created from workspace_principals.template.txt

   Please edit the principals file:
   /path/to/config/principals/analytics_dev_principals.txt

   Add user/group Object IDs (not emails!)

   Press ENTER after editing (or 's' to skip):
```

Option A: Add Users Now

1. Open a new terminal window
2. Edit the principals file:

   ```
   nano config/principals/analytics_dev_principals.txt
   ```

3. Add users (format: `principal_id,role,description,type`):

   ```
   9117cbfa-f0a7-43b7-846f-96ba66a3c1c0,Admin,John Doe,User
   a2b3c4d5-e6f7-8901-2345-6789abcdef01,Member,Analytics Team,Group
   ```

4. Save file (Ctrl+X, Y, Enter)
5. Return to first terminal and press **ENTER**

Option B: Skip and Add Later - Press `s` to skip - Add users manually later: `bash python ops/scripts/manage_workspaces.py add-users-from-file \ <workspace-id> \ config/principals/analytics_dev_principals.tx`

Automation Mode (CI/CD)

For non-interactive deployments:

```
python scenarios/config-driven-workspace/config_driven_workspace.py \
  --project analytics \
  --environment dev \
  --capacity-id $FABRIC_CAPACITY_ID \
```

```
  --principals-file config/principals/analytics_dev_principals.txt \
  --skip-user-prompt
```

Key flags: - `--skip-user-prompt`: No interactive prompts - `--principals-file`: Pre-created principals file

Verify Creation

Option 1: List Workspaces

```
python ops/scripts/manage_workspaces.py list
```

Option 2: Check Specific Workspace

```
python ops/scripts/manage_workspaces.py get --name "usf2-fabric-analytics-dev"
```

Option 3: View in Fabric Portal - Open: https://app.fabric.microsoft.com - Navigate to workspaces - Find: `usf2-fabric-analytics-dev`

Scenario 2: Domain-Workspace

Best for: Direct control over workspace names, simpler setup

Overview

This scenario lets you specify exact workspace names without config patterns:

```
Input: --domain-name customer-analytics
Output: customer-analytics-workspace (exact name you provide + "-workspace")
```

Navigate to Scenario

```
cd scenarios/domain-workspace
```

Basic Usage

```
python domain_workspace_with_existing_items.py \
  --domain-name customer-analytics \
  --capacity-id $FABRIC_CAPACITY_ID
```

What gets created: - Workspace: `customer-analytics-workspace` - Lakehouse: `CustomerAnalyticsLakehouse` - Warehouse: `CustomerAnalyticsWarehouse` - Staging Lakehouse: `CustomerAnalyticsStagingLakehouse` - Principals template

Console Output:

```
================================================================================
  Domain-Based Workspace Setup
================================================================================
```

```
  Configuration:
    Domain Name:       customer-analytics
    Workspace Name:    customer-analytics-workspace
    Capacity ID:       0749b635-c51b-46c6-948a-02f05d7fe177

======================================================================

  Workspace created: customer-analytics-workspace
  Lakehouse created: CustomerAnalyticsLakehouse
  Warehouse created: CustomerAnalyticsWarehouse
  Staging lakehouse created: CustomerAnalyticsStagingLakehouse
```

With Pre-Created Principals File

Step 1: Create principals file

```
cd ../..  # Return to project root
cp config/principals/workspace_principals.template.txt \
   config/principals/customer_analytics_principals.txt
```

Step 2: Edit file

```
nano config/principals/customer_analytics_principals.txt
```

Step 3: Add users

```
9117cbfa-f0a7-43b7-846f-96ba66a3c1c0,Admin,Administrator,User
b2c3d4e5-f6a7-8901-2345-6789abcdef12,Viewer,Analytics Team,Group
```

Step 4: Run scenario with principals

```
cd scenarios/domain-workspace
python domain_workspace_with_existing_items.py \
  --domain-name customer-analytics \
  --capacity-id $FABRIC_CAPACITY_ID \
  --principals-file ../../config/principals/customer_analytics_principals.txt \
  --skip-user-prompt
```

Verify Creation

```
# List all items in workspace
python ../../ops/scripts/manage_workspaces.py list-items <workspace-id>

# Expected output:
# - CustomerAnalyticsLakehouse (Lakehouse)
# - CustomerAnalyticsWarehouse (Warehouse)
# - CustomerAnalyticsStagingLakehouse (Lakehouse)
```

Scenario 3: Feature Branch Workflow

Best for: Ticket-based development with Git integration

Overview

Creates isolated feature workspaces linked to JIRA/ADO tickets:

```
Input: --feature JIRA-12345
Output:
  - Workspace: My Product [Feature JIRA-12345]
  - Git Branch: feature/my_product/JIRA-12345
  - Scaffold: data_products/my_product/
```

Prerequisites

Ensure Git working directory is clean:

```
git status
# Should show: nothing to commit, working tree clean
```

If you have uncommitted changes:

```
git add .
git commit -m "Your commit message"
```

Navigate to Scenario

```
cd scenarios/feature-branch-workflow
```

Step 1: Review Product Descriptor

```
cat product_descriptor.yaml
```

Or create your own:

```
cp product_descriptor.yaml my_product.yaml
nano my_product.yaml
```

Sample YAML structure:

```
product:
  name: "Customer Insights"
  description: "Customer analytics and insights platform"
  owner_email: "data-team@company.com"
  domain: "Customer Analytics"

environments:
  dev:
    enabled: true
    capacity_type: "trial"
```

```yaml
    description: "Development workspace for customer insights"

git:
  organization: "${GITHUB_ORG}"
  repository: "${GITHUB_REPO}"
  default_branch: "main"
  feature_prefix: "feature"
  directory: "data_products/customer_insights"
  auto_commit: true

scaffold:
  enabled: true
  directories:
    - "workspace"
    - "notebooks"
    - "pipelines"
    - "datasets"

  notebooks:
    - name: "ingestion_pipeline"
      language: "PySpark"
      description: "Customer data ingestion"
```

Step 2: Run with Feature Flag

```
python3 ../../ops/scripts/onboard_data_product.py \
  product_descriptor.yaml \
  --feature JIRA-12345
```

Console Output:

```
Loaded 1 environment variables from .env
Starting onboarding for product 'Customer Insights' (slug: customer_insights)
Seeded scaffold for customer_insights from template
Creating Fabric workspace: Customer Insights [DEV]
Created workspace 'Customer Insights [DEV]'
  ID: fc0f2e9d-dfee-4d50-9225-6f001c45abb6
  URL: https://app.fabric.microsoft.com/groups/fc0f2e9d-...

Creating feature workspace: Customer Insights [Feature JIRA-12345]
Created feature workspace 'Customer Insights [Feature JIRA-12345]'
  ID: 7306416a-339e-450c-995d-2d4162f7bbc0
  URL: https://app.fabric.microsoft.com/groups/7306416a-...

Creating git branch feature/customer_insights/JIRA-12345 from main
Git branch created: feature/customer_insights/JIRA-12345
Connected workspace to Git branch
```

```
Updated onboarding registry
Audit log written to .onboarding_logs/20251022_customer_insights_JIRA-12345.json

Onboarding complete! Feature workspace ready for development.
```

Step 3: Verify Creation

Check workspaces:

```
python ../../ops/scripts/manage_workspaces.py list | grep "Customer Insights"
```

Expected output:

```
Customer Insights [DEV]                    fc0f2e9d-dfee-4d50-9225-6f001c45abb6
Customer Insights [Feature JIRA-12345]     7306416a-339e-450c-995d-2d4162f7bbc0
```

Check Git branch:

```
git branch | grep "feature/customer_insights"
```

Expected output:

```
* feature/customer_insights/JIRA-12345
```

Check scaffold structure:

```
ls -la ../../data_products/customer_insights/
```

Expected output:

```
drwxr-xr-x datasets/
drwxr-xr-x docs/
drwxr-xr-x notebooks/
drwxr-xr-x pipelines/
drwxr-xr-x workspace/
-rw-r--r-- README.md
```

Step 4: Start Development

Work in your feature workspace via: 1. **Fabric Portal**: Open feature workspace and create items 2. **Local Git**: Edit files in `data_products/customer_insights/`

Step 5: Create Pull Request

After completing your work:

```
# Push feature branch
git push origin feature/customer_insights/JIRA-12345

# Create PR via GitHub CLI
gh pr create \
  --base main \
```

```
  --head feature/customer_insights/JIRA-12345 \
  --title "JIRA-12345: Add customer insights pipeline"

# Or create PR via GitHub web UI
```

Step 6: Cleanup After Merge

Once your PR is merged:

```
# Delete feature workspace (via Fabric portal or API)
python ../../ops/scripts/manage_workspaces.py delete \
  7306416a-339e-450c-995d-2d4162f7bbc0

# Delete Git branch
git checkout main
git branch -d feature/customer_insights/JIRA-12345
git push origin --delete feature/customer_insights/JIRA-12345
```

Comparison Matrix

Feature	Config-Driven	Domain-Workspace	Feature-Branch
Naming	Pattern-based	Direct	Product-based
Best For	Enterprise	Simple projects	Ticket development
Config File	project.config.json	None	product_descriptor.yaml
Git Integration	Optional	Optional	Built-in
Workspace Count	1 per run	1 per run	2 (DEV + Feature)
User Management	Built-in	Built-in	Manual
Cleanup	Manual	Manual	After merge

When to Use Each Scenario

Use Config-Driven When: - Large organization with naming standards - Multiple environments (DEV/TEST/PROD) - Governance and compliance requirements - Consistent naming across all workspaces

Use Domain-Workspace When: - Small team or single project - Want full control over names - Quick setup without config files - Domain-specific workspaces (Finance, Sales, HR)

Use Feature-Branch When: - Ticket-based development (JIRA, ADO) - Parallel development by multiple developers - Need isolated testing environments - Git integration for change tracking

---

Troubleshooting

Issue: ModuleNotFoundError

Symptom:

```
ModuleNotFoundError: No module named 'msal'
```

Solution:

```
pip install -r requirements.txt
```

Issue: 403 Forbidden (Lakehouse Creation)

Symptom:

```
Lakehouse creation failed: 403 Forbidden
```

Causes: 1. No capacity ID provided (using Trial workspace) 2. Invalid or inaccessible capacity ID 3. Service principal lacks permissions on capacity

Solution:

```
# Verify capacity ID is set
echo $FABRIC_CAPACITY_ID

# If empty, add to .env
nano .env
# Add: FABRIC_CAPACITY_ID=your-capacity-guid-here

# Reload environment
source .env

# Get capacity ID from Fabric portal:
# Settings → Admin portal → Capacity settings → Copy ID
```

Issue: Workspace Already Exists

Symptom:

```
Failed to create workspace: Workspace 'my-workspace' already exists
```

Solution Option 1: Delete existing workspace

```
# List workspaces to get ID
python ops/scripts/manage_workspaces.py list
```

```
# Delete specific workspace
python ops/scripts/manage_workspaces.py delete <workspace-id>
```

Solution Option 2: Use different name

```
# For config-driven: change project name
python scenarios/config-driven-workspace/config_driven_workspace.py \
  --project analytics-v2 \
  --environment dev

# For domain-workspace: change domain name
python scenarios/domain-workspace/domain_workspace_with_existing_items.py \
  --domain-name customer-analytics-v2
```

Issue: Authentication Failed

Symptom:

```
Authentication failed: Invalid credentials
```

Solution:

```
# Test authentication
python diagnostics/diagnose_fabric_permissions.py

# Check Azure AD token
python diagnostics/check_graph_permissions.py

# Clear cached tokens
python diagnostics/clear_token_cache.py

# Verify .env credentials are correct
cat .env | grep -E "CLIENT_ID|TENANT_ID|CLIENT_SECRET"
```

Issue: Git Working Directory Not Clean

Symptom:

```
Cannot create feature branch: working directory has uncommitted changes
```

Solution:

```
# Check status
git status

# Commit changes
git add .
git commit -m "Prepare for feature branch"
```

```
# Or stash changes temporarily
git stash
# ... run scenario ...
git stash pop
```

Issue: Import Errors

Symptom:

```
ImportError: cannot import name 'WorkspaceManager'
```

Solution:

```
# Verify Python path
python -c "import sys; print('\n'.join(sys.path))"

# Ensure you're in project root
pwd
# Should show: .../usf-fabric-cicd

# Reinstall dependencies
pip install --force-reinstall -r requirements.txt
```

Next Steps

After Successful First Run

1. **View in Fabric Portal**
   - Navigate to: https://app.fabric.microsoft.com
   - Find your workspace
   - Explore created items
2. **Review Setup Logs**

   ```
   cat config/setup-logs/analytics_dev_setup_log.json | jq '.'
   ```
3. **Add More Users**

   ```
   # Edit principals file
   nano config/principals/analytics_dev_principals.txt

   # Add users via CLI
   python ops/scripts/manage_workspaces.py add-users-from-file \
     <workspace-id> \
     config/principals/analytics_dev_principals.txt
   ```
4. **Create Additional Environments**

```
# Create TEST environment
python scenarios/config-driven-workspace/config_driven_workspace.py \
  --project analytics \
  --environment test \
  --capacity-id $FABRIC_CAPACITY_ID

# Create PROD environment
python scenarios/config-driven-workspace/config_driven_workspace.py \
  --project analytics \
  --environment prod \
  --capacity-id $FABRIC_CAPACITY_ID
```

5. **Explore Other Scenarios**

```
ls scenarios/
# Try: domain-workspace, feature-branch-workflow, leit-ricoh-setup
```

Learning Resources

- **Microsoft Fabric Documentation**: https://learn.microsoft.com/fabric/
- **Fabric REST API Reference**: https://learn.microsoft.com/rest/api/fabric/
- **Project Documentation**: Check `docs/` directory
- **Scenario-Specific Guides**: `scenarios/*/README.md`

Getting Help

Check Documentation:

```
# Main project README
cat README.md

# Scenario-specific help
cat scenarios/config-driven-workspace/README.md
cat scenarios/domain-workspace/README.md
cat scenarios/feature-branch-workflow/README.md
```

Run Diagnostics:

```
./setup/preflight_check.sh
python diagnostics/diagnose_fabric_permissions.py
```

Appendix A: Common Commands Reference

Workspace Management

```
# List all workspaces
python ops/scripts/manage_workspaces.py list
```

```
# Get workspace details
python ops/scripts/manage_workspaces.py get --name "workspace-name"

# Delete workspace
python ops/scripts/manage_workspaces.py delete <workspace-id>

# List workspace items
python ops/scripts/manage_workspaces.py list-items <workspace-id>
```

User Management

```
# Add single user
python ops/scripts/manage_workspaces.py add-user \
  <workspace-id> <user-object-id> --role Admin

# Add users from file
python ops/scripts/manage_workspaces.py add-users-from-file \
  <workspace-id> config/principals/users.txt

# Preview before adding (dry-run)
python ops/scripts/manage_workspaces.py add-users-from-file \
  <workspace-id> config/principals/users.txt --dry-run
```

Diagnostics

```
# Check Fabric permissions
python diagnostics/diagnose_fabric_permissions.py

# Check Graph API permissions
python diagnostics/check_graph_permissions.py

# Clear authentication cache
python diagnostics/clear_token_cache.py

# Verify workspace exists
python diagnostics/verify_workspace.py <workspace-id>
```

Environment Management

```
# Activate conda environment
conda activate fabric-cicd

# Update dependencies
pip install --upgrade -r requirements.txt

# Check environment status
./setup/preflight_check.sh
```

```
# Deactivate environment
conda deactivate
```

Appendix B: File Locations Reference

Configuration Files

```
project.config.json              # Project naming patterns
.env                             # Azure credentials (DO NOT COMMIT)
.env.example                     # Template for .env
requirements.txt                 # Python dependencies
```

Principals Files

```
config/principals/
  workspace_principals.template.txt    # Template
  {project}_{env}_principals.txt       # Generated per workspace
  custom_team.txt                      # Custom principal files
```

Setup Logs

```
config/setup-logs/
  {project}_{env}_setup_log.json       # Generated per workspace
```

Scenarios

```
scenarios/
  config-driven-workspace/             # Config-based naming
  domain-workspace/                    # Direct naming
  feature-branch-workflow/             # Git-integrated
  leit-ricoh-setup/                   # Legacy LEIT-Ricoh
```

Utilities

```
ops/scripts/
  manage_workspaces.py                 # Workspace CLI
  deploy_fabric.py                     # Deployment tool
  utilities/
      workspace_manager.py             # Workspace API
      fabric_item_manager.py           # Item API
      config_manager.py                # Config handling
```

Document Change Log

Version	Date	Author	Changes
1.0	Oct 2024	Platform Team	Initial release

End of Document

For the latest version, visit: [Internal Documentation Portal]