GitHub Copilot Tool Sets 🧰

Tool Sets extend GitHub Copilot's capabilities by integrating external tools, APIs, and services directly into your AI workflow. They enable Copilot to perform actions beyond code generation, such as running tests, deploying applications, and interacting with databases.

🎯 What Are Tool Sets?

Tool Sets allow Copilot to:

- **Execute Commands**: Run terminal commands and scripts
- **API Integration**: Connect with external services and APIs
- **Database Operations**: Query and manipulate database data
- **File Operations**: Read, write, and manipulate files
- **Testing**: Run automated tests and analyze results
- **Deployment**: Deploy applications to cloud platforms

Built-in vs Custom Tool Sets

Built-in Tool Sets	Custom Tool Sets
Terminal, Git, npm/Maven	Database connectors, Cloud APIs
Testing frameworks	Organization tools
File system operations	Custom scripts and utilities
Basic HTTP requests	Third-party service integrations

🚀 Getting Started with Tool Sets

Enabling Tool Sets

```
# Via Settings
"github.copilot.toolsets.enabled": true
```

```
# Via Command Palette
Ctrl/Cmd + Shift + P → "Copilot: Enable Tool Sets"

# Per-workspace configuration
// .vscode/settings.json
{
  "github.copilot.toolsets": {
    "terminal": true,
    "git": true,
    "npm": true,
    "maven": true,
    "database": false
  }
}
```

Basic Tool Set Usage

```
# In Copilot Chat
/tools run "mvn clean test"
/tools git status
/tools npm install lodash
/tools query SELECT * FROM users WHERE active = true
```

🛠️ Built-in Tool Sets

Terminal Tool Set 💻

Capabilities: Execute command-line operations

```
# Examples
/tools terminal "ls -la src/"
/tools run "java --version"
/tools exec "find . -name '*.java' | grep Controller"

# Multi-step operations
/tools terminal "cd project1/task-manager && mvn clean && mvn compile"
```

Configuration:

```
{
  "github.copilot.toolsets.terminal": {
    "enabled": true,
    "allowedCommands": ["mvn", "npm", "git", "java", "python"],
    "workingDirectory": "auto",
    "timeout": 30000
  }
}
```

Git Tool Set 📝

Capabilities: Version control operations

```
# Examples
/tools git status
/tools git "add src/main/java/com/example/Controller.java"
/tools git "commit -m 'Add user controller'"
/tools git "log --oneline -5"

# Advanced operations
/tools git "diff HEAD~1 src/main/java/com/example/"
/tools git "branch -a | grep feature"
```

Best Practices:

```
✅ Use for: Status checks, simple commits, branch operations
❌ Avoid for: Complex merges, interactive rebases, sensitive operations
```

Maven Tool Set 📦

Capabilities: Java project build and dependency management

```
# Examples
/tools maven "clean compile"
/tools mvn "test -Dtest=UserServiceTest"
/tools maven "dependency:tree"
/tools mvn "spring-boot:run"

# Analysis operations
/tools maven "help:effective-pom"
/tools mvn "versions:display-dependency-updates"
```

Configuration:

```
{
  "github.copilot.toolsets.maven": {
    "enabled": true,
    "defaultGoals": ["clean", "compile", "test"],
    "profiles": ["dev", "test", "prod"],
    "jvmArgs": "-Xmx2g"
  }
}
```

NPM Tool Set 📦

Capabilities: Node.js package management

```
# Examples
/tools npm "install express"
/tools npm "run test"
/tools npm "audit fix"
/tools npm "ls --depth=0"

# Development operations
/tools npm "run dev"
/tools npm "run build"
/tools npm "run lint"
```

Testing Tool Set 🧪

Capabilities: Test execution and analysis

```
# Examples
/tools test "run UserServiceTest"
/tools test "mvn test -Dtest=*Controller*"
/tools test "npm test -- --coverage"

# Analysis
/tools test "analyze coverage report"
/tools test "find failing tests"
```

Configuration:

```
{
  "github.copilot.toolsets.testing": {
    "frameworks": ["junit", "testng", "jest", "mocha"],
    "coverage": {
      "enabled": true,
      "threshold": 80
    },
    "reports": {
      "format": ["json", "html"],
      "location": "target/test-reports"
    }
  }
}
```

🏢 Custom Tool Sets

Database Tool Set 🗄️

Setup:

```
// .vscode/settings.json
{
  "github.copilot.toolsets.database": {
    "enabled": true,
    "connections": {
      "development": {
        "type": "h2",
        "url": "jdbc:h2:mem:testdb",
        "username": "sa",
        "password": ""
      },
      "testing": {
        "type": "postgresql",
        "host": "localhost",
        "port": 5432,
        "database": "testdb"
      }
    }
  }
}
```

Usage:

```
# Examples
/tools db "SELECT COUNT(*) FROM users"
/tools query "SHOW TABLES"
/tools db "INSERT INTO users (name, email) VALUES ('John',
'john@example.com')"

# Analysis
/tools db "EXPLAIN SELECT * FROM users WHERE email LIKE '%@example.com'"
```

Cloud Platform Tool Set ☁️

AWS Tool Set:

```
{
  "github.copilot.toolsets.aws": {
    "enabled": true,
    "services": ["s3", "ec2", "lambda", "rds"],
    "region": "us-east-1",
    "profile": "development"
  }
}
```

Usage:

```
# Examples
/tools aws "s3 ls my-bucket"
/tools aws "ec2 describe-instances"
/tools aws "lambda list-functions"

# Deployment
/tools aws "sam deploy --stack-name my-app"
```

Docker Tool Set 🐳

Configuration:

```
{
  "github.copilot.toolsets.docker": {
    "enabled": true,
    "registries": ["docker.io", "ghcr.io"],
    "buildArgs": {
      "JAVA_VERSION": "21",
      "MAVEN_VERSION": "3.9"
    }
  }
}
```

Usage:

```
# Examples
/tools docker "build -t my-app:latest ."
/tools docker "run -p 8080:8080 my-app:latest"
/tools docker "ps -a"
```

```
/tools docker "logs container-name"

# Multi-container operations
/tools docker "compose up -d"
/tools docker "compose logs web"
```

💡 Advanced Tool Set Techniques

Chained Tool Operations

```
# Sequential execution
/tools maven "clean compile" && /tools test "run integration tests" && /
tools docker "build -t app:latest ."

# Conditional execution
/tools git "status" | if clean then /tools git "pull origin main"

# Pipeline operations
/tools maven "package" | /tools docker "build -t myapp ." | /tools aws
"ecr push myapp"
```

Context-Aware Tool Usage

```
# Use current file context
/tools test #file:UserController.java "run tests for this controller"

# Use selection context
/tools git #selection "add these specific lines to staging"

# Use project context
/tools maven #file:pom.xml "analyze dependencies and suggest updates"
```

Custom Tool Scripts

```
# .copilot/tools/deploy.sh
#!/bin/bash
echo "Deploying application..."
mvn clean package
docker build -t myapp:$1 .
docker push myapp:$1
kubectl set image deployment/myapp myapp=myapp:$1
```

Usage:

```
/tools script "deploy.sh v1.2.3"
```

🎨 Best Practices

✅ Do's

1. **Start with Built-in Tools**: Master basic tool sets first `markdown # Good progression /tools git status # Basic git operations /tools maven clean test # Build and test /tools terminal "ls -la" # File system exploration`
2. **Verify Tool Availability**: Check tool installation `markdown /tools terminal "mvn --version" /tools terminal "java --version" /tools terminal "git --version"`
3. **Use Appropriate Context**: Include relevant files and selections `markdown /tools test #file:pom.xml "run tests for this project configuration"`
4. **Chain Related Operations**: Group logical operations `markdown /tools maven "clean" && /tools maven "compile" && /tools test "run unit tests"`

❌ Don'ts

1. **Don't Execute Dangerous Commands**: Avoid destructive operations ```markdown # Dangerous - avoid /tools terminal "rm -rf /" /tools git "reset --hard HEAD~10" /tools db "DROP DATABASE production"

# Safe alternatives /tools git "status" /tools db "SELECT COUNT(*) FROM users" ```

1. **Don't Ignore Security**: Be cautious with credentials and sensitive data ```markdown # Bad - exposes credentials /tools db "mysql -u root -p secret123 -e 'SELECT * FROM users'"

# Good - use configuration /tools db "SELECT * FROM users" # Uses configured connection ```

1. **Don't Skip Validation**: Always verify command results ```markdown /tools maven "test" # Always check: Did tests pass? Any failures?

/tools git "commit -m 'Fix bug'" # Always verify: Was commit successful? ```

🚨 Common Pitfalls

Permission Issues

Problem: Tool execution fails due to permissions **Solution**: Check file permissions and user privileges

```
# Diagnosis
/tools terminal "ls -la script.sh"
# Fix
/tools terminal "chmod +x script.sh"
```

Path Problems

Problem: Tools not found in PATH **Solution**: Use full paths or configure environment

```
# Problem
/tools terminal "mvn clean"  # Command not found

# Solution
/tools terminal "/usr/local/bin/mvn clean"
# Or configure PATH in settings
```

Resource Conflicts

Problem: Multiple tools competing for resources **Solution**: Serialize operations and check resource usage

```
# Avoid parallel resource-intensive operations
/tools maven "clean install" # Wait for completion
/tools test "run all tests"   # Then run tests
```

🔧 Troubleshooting Tool Sets

Tool Set Not Working

1. **Check Tool Set Status** `markdown /tools status /tools list available`

2. **Verify Configuration** `json { "github.copilot.toolsets.enabled": true, "github.copilot.toolsets.terminal.enabled": true }`

3. **Test Individual Tools** `markdown /tools terminal "echo 'Hello World'" /tools git "version"`

Poor Performance

1. **Optimize Tool Configuration** `json { "github.copilot.toolsets.terminal.timeout": 10000, "github.copilot.toolsets.maven.maxMemory": "1g" }`

2. **Use Selective Tool Sets** `json { "github.copilot.toolsets": { "terminal": true, "git": true, "maven": true, "database": false, // Disable unused tools "aws": false } }`

🎓 Learning Exercises

Beginner

1. **Basic Tool Operations** `markdown /tools git status /tools terminal "pwd" /tools maven "clean"`

2. **File System Exploration** `markdown /tools terminal "find . -name '*.java'" /tools terminal "grep -r 'TODO' src/"`

Intermediate

1. **Build and Test Workflows** `markdown /tools maven "clean compile test" /tools terminal "java -jar target/app.jar"`

2. **Database Interactions** `markdown /tools db "SELECT * FROM information_schema.tables" /tools db "SHOW PROCESSLIST"`

Advanced

1. **Custom Tool Integration** `markdown Create custom deployment scripts Configure cloud platform tools Set up automated testing pipelines`

2. **Tool Set Optimization** `markdown Profile tool performance Configure tool-specific settings Create tool usage guidelines`

Remember: Tool Sets are powerful extensions to Copilot's capabilities. Start with built-in tools, understand security implications, and gradually introduce custom tools as your needs grow.