

# lab-09-capstone-build-agent

- [Lab 09: Capstone - Build a Production-Ready Agent](#)
  - [Objectives](#)
  - [Prerequisites](#)
  - [Overview](#)
  - [Step 1: Select Your Agent Role \(5 minutes\)](#)
    - [Option A: Code Reviewer](#)
    - [Option B: Documentation Writer](#)
    - [Option C: Performance Auditor](#)
    - [Option D: Security Reviewer](#)
    - [Option E: Your Custom Agent](#)
    - [Decision](#)
  - [Step 2: Define Success Criteria \(5 minutes\)](#)
    - [Success Criteria Template](#)
    - [Test Scenarios](#)
  - [Step 3: Create the Agent Definition \(15 minutes\)](#)
    - [Instructions](#)
  - [Step 4: Test Your Agent \(10 minutes\)](#)
    - [Test Procedure](#)
    - [Test Results Template](#)
  - [Step 5: Iterate and Refine \(5 minutes\)](#)
    - [Common Issues and Fixes](#)
    - [Refinement Process](#)
  - [Step 6: Document Your Agent \(5 minutes\)](#)
    - [Agent Documentation Template](#)
  - [Deliverables](#)
  - [Group Share \(If in Workshop Setting\)](#)
  - [Key Takeaways](#)
  - [Next Steps: Taking Agents to Production](#)
    - [Before sharing your agent with the team:](#)
    - [Continuous Improvement](#)
  - [Bonus Challenge \(Optional\)](#)
  - [Workshop Conclusion \(Module 6\)](#)
    - [What You've Learned](#)
    - [Putting It Into Practice](#)
    - [Final Reflection](#)
  - [Additional Resources](#)

## Lab 09: Capstone - Build a Production-Ready Agent

**Module:** 5

**Duration:** 35 minutes

**Part:** Advanced GitHub Copilot (Part 2)

# Objectives

By the end of this lab, you will:

- Create a functional custom agent from scratch
- Apply all design principles learned in previous labs
- Test and iterate on agent behavior
- Prepare an agent for team use with clear documentation

## Prerequisites

- Completion of [Lab 08: Agent Design](#)
- VS Code with GitHub Copilot extension
- Access to the TaskManager workshop repository
- Understanding of agent components and design patterns

## Overview

In this capstone lab, you'll **build your own custom agent** aligned to a real workflow. You'll define success criteria, create the agent definition, test it, iterate, and prepare it for team use.

---

## Step 1: Select Your Agent Role (5 minutes)

Choose **one** of these agent roles (or propose your own):

### Option A: Code Reviewer

**Role:** Reviews code for quality, standards, and best practices

**Workflow:** Pull request reviews, pre-commit checks

**Value:** Consistent code quality across team members

### Option B: Documentation Writer

**Role:** Generates or reviews documentation for clarity and completeness

**Workflow:** API docs, README updates, architectural decision records

**Value:** Consistent documentation standards

### Option C: Performance Auditor

**Role:** Identifies performance issues and optimization opportunities

**Workflow:** Reviewing slow endpoints, database queries, algorithms

**Value:** Proactive performance monitoring

### Option D: Security Reviewer

**Role:** Reviews code for security vulnerabilities and best practices

**Workflow:** Pre-deployment security checks, sensitive data handling

**Value:** Reduced security risks

## Option E: Your Custom Agent

**Role:** [Define your own based on team needs]

**Workflow:** [Describe the workflow]

**Value:** [What problem does it solve?]

### Decision

**I will create:** [Agent name and role]

---

## Step 2: Define Success Criteria (5 minutes)

Before writing instructions, define what "good" looks like for your agent.

### Success Criteria Template

**My agent is successful when:**

1. [Criterion 1 - e.g., "It identifies all security vulnerabilities in sample code"]
2. [Criterion 2 - e.g., "It provides actionable remediation steps"]
3. [Criterion 3 - e.g., "It doesn't flag false positives for safe patterns"]
4. [Criterion 4 - e.g., "Output is structured and consistent"]
5. [Criterion 5 - e.g., "It can be used by any team member with the same results"]

### Test Scenarios

**I will test my agent with:**

1. [Test scenario 1 - e.g., "Code with SQL injection vulnerability"]
  2. [Test scenario 2 - e.g., "Code with proper input validation"]
  3. [Test scenario 3 - e.g., "Edge case or boundary condition"]
- 

## Step 3: Create the Agent Definition (15 minutes)

Create a new file: .github/agents/[your-agent-name].agent.md

Use this template and customize for your agent:

```
---  
name: "[agent-name]"  
description: '[Brief description of what this agent does]'  
tools: [changes] # or [] if no tools needed  
model: Claude Sonnet 4  
---  
  
# [Agent Name]
```

You are an expert [role/specialty] with deep knowledge of [domain expertise].

## ## Responsibilities

- [Responsibility 1]
- [Responsibility 2]
- [Responsibility 3]
- [Responsibility 4]

## ## Context

This project follows [relevant context about the codebase, standards, patterns]:

- [Context point 1]
- [Context point 2]
- [Context point 3]

## ## Constraints

- ALWAYS [critical rule the agent must follow]
- NEVER [something the agent must avoid]
- [Additional constraint]
- [Additional constraint]

## ## Analysis Process

1. [Step 1 of how the agent should approach the task]
2. [Step 2]
3. [Step 3]
4. [Step 4]

## ## Output Format

Provide your [review/analysis/report] in this structured format:

### ### [Section 1 Title]

- \*\*[Field]\*\*: [description]
- \*\*[Field]\*\*: [description]

### ### [Section 2 Title]

[Description of what goes in this section]

### ### [Section 3 Title]

[Description of what goes in this section]

## ## Tone

- [Tone guideline 1 - e.g., "Be direct and constructive"]
- [Tone guideline 2 - e.g., "Explain WHY, not just WHAT"]

- [Tone guideline 3 - e.g., "Acknowledge good practices"]

## ## Examples of What to Flag

- [Example 1]
- [Example 2]
- [Example 3]
- [Example 4]

## Instructions

1. Create the file .github/agents/[your-agent-name].agent.md
  2. Fill in all sections based on your chosen role
  3. Be specific in constraints and output format
  4. Include examples where helpful
- 

## Step 4: Test Your Agent (10 minutes)

### Test Procedure

1. **Open Copilot Chat in Agent Mode**
2. **Select your custom agent** from the dropdown
3. **Run each test scenario** you defined in Step 2
4. **Record the results:**
  - Did the agent follow the output format?
  - Did it meet your success criteria?
  - Were there unexpected behaviors?
  - What worked well? What didn't?

### Test Results Template

#### Test 1: [Scenario description]

- **Agent Output:** [Summary or excerpt]
- **Success Criteria Met:** [Yes/No - which ones?]
- **Issues Found:** [List any problems]

#### Test 2: [Scenario description]

- **Agent Output:** [Summary or excerpt]
- **Success Criteria Met:** [Yes/No - which ones?]
- **Issues Found:** [List any problems]

#### Test 3: [Scenario description]

- **Agent Output:** [Summary or excerpt]
  - **Success Criteria Met:** [Yes/No - which ones?]
  - **Issues Found:** [List any problems]
-

## Step 5: Iterate and Refine (5 minutes)

Based on your test results, refine your agent:

### Common Issues and Fixes

Issue	Likely Cause	Fix
Output format inconsistent	Vague instructions	Add explicit structure with headings
Agent goes out of scope	No constraints	Add "NEVER" constraints
Too verbose	No guidance on brevity	Add tone guidance: "Be concise"
Misses important checks	Missing from responsibilities	Add to responsibilities list
False positives	Overly broad rules	Add examples of acceptable patterns

### Refinement Process

1. Identify the top 2-3 issues from testing
  2. Update your agent's instructions
  3. **Re-test** with the same scenarios
  4. Repeat until success criteria are met
- 

## Step 6: Document Your Agent (5 minutes)

Create a brief usage guide for your agent:

### Agent Documentation Template

**Agent Name:** [Your agent name]

**Purpose:** [One-sentence description]

#### When to Use:

- [Use case 1]
- [Use case 2]
- [Use case 3]

#### When NOT to Use:

- [Anti-pattern 1]
- [Anti-pattern 2]

#### Example Prompts:

[Example prompt 1]  
[Example prompt 2]  
[Example prompt 3]

#### Expected Output:

- [What users should expect]
- [Output structure/format]

## **Limitations:**

- [Limitation 1]
- [Limitation 2]

## **Maintenance:**

- **Owner:** [Your name or team]
  - **Last Updated:** [Date]
  - **Review Frequency:** [e.g., Quarterly]
- 

## **Deliverables**

At the end of this lab, you should have:

- A custom agent definition file (.github/agents/[name].agent.md)
  - Test results showing the agent meets success criteria
  - Usage documentation for team members
  - At least one iteration/refinement based on testing
- 

## **Group Share (If in Workshop Setting)**

### **Demonstrate your agent:**

1. Show the agent name and purpose
2. Run a live demo with a test scenario
3. Show the structured output
4. Share one key design decision you made
5. Share one challenge you encountered

### **Learn from others:**

- What agents did others create?
  - What patterns emerged across different agents?
  - What would you borrow for your next agent?
- 

## **Key Takeaways**

- Start with success criteria** - Define "done" before building
  - Test early and often** - Don't wait until it's "perfect"
  - Iterate based on real usage** - Agents improve over time
  - Document for others** - Agents are team assets, not personal tools
  - Keep scope focused** - Better to excel at one thing than be mediocre at many
-

# Next Steps: Taking Agents to Production

**Before sharing your agent with the team:**

1. **Test with real scenarios** (not just examples)
2. **Get peer review** (have a colleague test it)
3. **Document edge cases** and limitations
4. **Add to the team catalog** ([docs/guides/custom-agent-catalog.md](#))
5. **Set up governance** (review process, versioning)

## Continuous Improvement

- Collect feedback from team usage
- Track common issues or misunderstandings
- Update instructions based on lessons learned
- Retire or merge agents that become obsolete

---

## Bonus Challenge (Optional)

**Advanced Exercise:**

Create a **second agent** that complements your first one. For example:

- If you built a Security Reviewer, build a Security Remediation Guide
- If you built a Documentation Writer, build a Documentation Reviewer
- If you built a Performance Auditor, build a Performance Optimizer

**Workflow:** Use both agents in sequence to demonstrate a complete workflow.

---

## Workshop Conclusion (Module 6)

Congratulations! You've completed the Advanced GitHub Copilot workshop.

### What You've Learned

- Ask, Edit, and Agent interaction models
- When and how to use custom agents
- How to design reliable, role-based agents
- Iteration and governance for agent maintenance
- Hands-on experience building a production-ready agent

### Putting It Into Practice

**Week 1:** Use existing agents (Architecture Reviewer, Backlog Generator, Test Strategist) in your daily work

**Week 2:** Identify one repetitive workflow and draft an agent for it

**Week 3:** Test and refine your agent with real scenarios

**Week 4:** Share with your team and gather feedback

## Final Reflection

1. Which workflows will benefit most from agents in your work?
  2. What agents should be standardized across your team?
  3. How will you prevent "prompt sprawl" as agents proliferate?
  4. Who will own agent governance on your team?
- 

## Additional Resources

- [Agent Design Guide](#)
  - [Agent Governance](#)
  - [Custom Agent Catalog](#)
  - [GitHub Documentation: Custom Agents](#)
- 

Thank you for participating in Advanced GitHub Copilot!

Questions or feedback? [Open an issue](#) or reach out to the workshop facilitators.