Lab 106: Microsoft Agent Framework : patrons d'orchestration
Level: L300 Parcours: ai-agents Durée: ~90 min Coût: free
info "Phase 3 · local first" Le parcours obligatoire est déterministe et local. Le parcours Réel est facultatif, estimé et nettoyé.

Ce que vous apprendrez

Implémenter séquentiel, concurrent, handoff et groupe Utiliser arrêt et escalade explicites Mesurer le coût de coordination Choisir le patron minimal suffisant

Prérequis

Python 3.10+ et un terminal Aucun identifiant pour le parcours obligatoire Lisez la spécification officielle avant un adaptateur hébergé

Modèle visuel

[diagram or code sample]

Scénario

OutdoorGear a besoin d'une implémentation révisable pour atelier, avec fixtures déterministes, frontières de confiance explicites, critères mesurables et adaptateur hébergé facultatif et réversible.

Exercice guidé

Step 1: sequential

1. Exécutez la fixture déterministe de cette étape. 2. Notez entrée, sortie, décision de politique et comportement d'échec. 3. Modifiez un comportement borné et relancez le test local.

Evidence: Conservez la commande, le résultat et une phrase expliquant pourquoi la preuve est fiable.

Step 2: concurrent

1. Exécutez la fixture déterministe de cette étape. 2. Notez entrée, sortie, décision de politique et comportement d'échec. 3. Modifiez un comportement borné et relancez le test local.

Evidence: Conservez la commande, le résultat et une phrase expliquant pourquoi la preuve est fiable.

Step 3: handoff

1. Exécutez la fixture déterministe de cette étape. 2. Notez entrée, sortie, décision de politique et comportement d'échec. 3. Modifiez un comportement borné et relancez le test local.

Evidence: Conservez la commande, le résultat et une phrase expliquant pourquoi la preuve est fiable.

Step 4: decision_record

1. Exécutez la fixture déterministe de cette étape. 2. Notez entrée, sortie, décision de politique et comportement d'échec. 3. Modifiez un comportement borné et relancez le test local.

Evidence: Conservez la commande, le résultat et une phrase expliquant pourquoi la preuve est fiable.

Exécuter localement

[diagram or code sample]

Simulation / Parcours Réel

Simulation Parcours Réel

Utilisez les fixtures déterministes et le validateur local. Coût estimé : 0 USD.

Remplacez uniquement la frontière de l'adaptateur. Fixez un plafond, gardez les mêmes tests et notez région, rétention, identité, télémétrie et prix. Supprimez déploiements, points de

terminaison, index, traces et données de test après validation.
Kit d'accompagnement
Le build du site génère des téléchargements PDF et ZIP stables. La CI conserve aussi un miroir temporaire pendant 30 jours.
README.md : démarrage et preuves CHEATSHEET.md : guide de décision scenario.json : contrat déterministe test_scenario.py : tests locaux Télécharger le PDF généré Télécharger le ZIP généré
Quiz noté
question "Pourquoi garder un contrat local ?"
??? success "Réponse" Il fournit une base déterministe avant la variabilité et le coût hébergés.
question "Quand le parcours Réel est-il terminé ?"
??? success "Réponse" Après enregistrement des preuves de validation et de nettoyage.
question "Comment traiter la sortie des outils ?"
??? success "Réponse" Comme donnée non fiable avant validation de politique et de schéma.
Critères de sortie
[ ] Les tests locaux passent sans identifiants. [ ] Le diagramme a une explication textuelle et ne dépend pas de la couleur. [ ] Le quiz noté a été revu. [ ] Le parcours Réel a un plafond et un plan de nettoyage. [ ] Sources et stabilité ont été vérifiées.
Sources publiques officielles
Microsoft Agent Framework 1.0 GA AutoGen to Agent Framework migration

Supporting file: CHEATSHEET.md
Microsoft Agent Framework : patrons d'orchestration ? Cheat Sheet
Commencez par le contrat local et des critères explicites. Traitez prompts, contenu récupéré, outils et messages comme non fiables. Rendez identité, autorisation, politique, télémétrie, coût et nettoyage explicites. Préférez l'orchestration ou l'adaptation minimale suffisante.

Supporting file: README.md
Microsoft Agent Framework : patrons d'orchestration
python -m unittest discover -v
Le parcours obligatoire est local et déterministe.

Supporting file: scenario.json
{ "lab": "106", "steps": ["sequential", "concurrent", "handoff", "decision_record"], "required_route": "local", "credentials_required": false, "real_path": { "optional": true, "cost_cap_required": true, "cleanup_required": true } }

Supporting file: test_scenario.py
import json import unittest from pathlib import Path
class Phase3ScenarioTests(unittest.TestCase): @classmethod def setUpClass(cls): cls.data = json.loads(Path("scenario.json").read_text(encoding="utf-8"))
def test_identity_and_steps(self): self.assertEqual(self.data["lab"], "106")
self.assertEqual(len(self.data["steps"]), 4)
def test_free_first_contract(self): self.assertEqual(self.data["required_route"], "local")
self.assertFalse(self.data["credentials_required"])
def test_real_path_guardrails(self): real = self.data["real_path"]

self.assertTrue(real["optional"]) self.assertTrue(real["cost_cap_required"])
self.assertTrue(real["cleanup_required"])
if __name__ == "__main__": unittest.main()