Cours 7

Model Context Protocol: Sujets Avancés

Anthropic Academy — Resume detaille

Yoann Boulch | 01/04/2026

1. Sampling: Serveurs appelant le modèle

■ Les serveurs peuvent demander au modèle d'effectuer des calculs via sampling

Le sampling permet aux serveurs MCP de demander au client d'utiliser le modèle IA pour accomplir une tâche. C'est l'inverse du flux normal où c'est le modèle qui appelle les outils.

Cas d'usage:

- Traitement de contenu avant d'exposer des ressources
- Validation et conformité des données
- Optimisation de requêtes
- Enrichissement de données avec l'IA
- Classification et triage automatique

2. Implémentation du sampling

■ Utilisez ctx.session.create_message pour invoquer le modèle

```
@mcp.tool() async def analyser_avec_llm(texte: str) -> str: """Utilise le modèle pour analyser du texte""" response = await ctx.session.create_message( model="claude-3-5-sonnet", max_tokens=1024, messages=[{ "role": "user", "content": f"Analysez: {texte}" }] ) return response.content[0].text
```

3. Logs et notifications de progression

■ Gardez l'utilisateur informé avec des messages de progression

MCP fournit des fonctions pour envoyer des informations, avertissements et rapports de progression.

```
@mcp.tool() def traiter_donnees(taille: int) -> str: ctx.info(f"Début du traitement de {taille} items") for i in range(taille): if i % 100 == 0: ctx.report_progress(i, taille) # traitement... ctx.info("Traitement terminé") return "Succès"
```

4. Système de permissions (Roots)

■ Les racines contrôlent l'accès au système de fichiers de manière sécurisée

Le système Roots permet aux applications de donner au serveur MCP l'accès à certains répertoires du système de fichiers, avec contrôle granulaire.

Avantages:

- Sécurité: limite l'accès à des répertoires spécifiques
- Auditabilité: suivi de qui accède à quoi
- Flexibilité: accordez des permissions dynamiquement

5. Types de messages JSON MCP

■ MCP utilise un format JSON standardisé pour tous les messages

Type	Flux	Description
Request	Client → Serveur	Appel d'outil ou accès ressource
Result	Serveur → Client	Réponse réussie
Notification	Bidirectionnel	Événement ou changement
Error	Serveur → Client	Erreur d'exécution

6. Transport STDIO

■ STDIO utilise l'entrée/sortie standard pour la communication locale

Le transport STDIO est idéal pour les serveurs MCP locaux qui communiquent avec un processus parent.

Caractéristiques:

- Communication locale (même machine)
- Pas de réseau, ultra-rapide
- Parfait pour les outils CLI
- Gestion des processus enfants

7. Transport StreamableHTTP (SSE)

■ HTTP avec Server-Sent Events pour la communication distribuée

StreamableHTTP utilise les Server-Sent Events (SSE) pour permettre à des serveurs MCP de communiquer sur le réseau avec des connexions persistantes.

```
# Configuration du serveur HTTP mcp_server = FastMCP("mon_api") app =
StreamableHTTPServer(mcp_server) # Servir sur le réseau if __name__ == "__main__":
app.run(host="0.0.0.0", port=8000)
```

8. État et scaling horizontal

■ Concevez les serveurs pour la scalabilité horizontale

Mode	État	Scaling	Cas d'usage
Stateless	Pas d'état local	Horizontal	APIs publiques
Stateful	État en mémoire	Vertical	Requêtes liées

9. Choisir entre STDIO et HTTP

■ STDIO pour local, HTTP pour distribué

Critère	STDIO	HTTP
Distance	Local	Réseau
Complexité	Basse	Modérée
Latence	Minimale	Normale
Scaling	Vertical	Horizontal
Sécurité	Implicite	À configurer

10. Intégration avec Pydantic

■ Validez les types de données avec Pydantic

Pydantic offre une validation robuste des données pour les serveurs MCP, générant automatiquement des schémas JSON corrects.

```
from pydantic import BaseModel from mcp.server.fastmcp import FastMCP class UserData(BaseModel):
nom: str age: int email: str mcp = FastMCP("user_api") @mcp.tool() def creer_utilisateur(data:
UserData) -> dict: """Crée un nouvel utilisateur""" return {"id": 123, "user": data}
```

11. Déploiement et infrastructure

■ Déployez les serveurs MCP en production

Les serveurs MCP peuvent être déployés de plusieurs manières selon l'architecture souhaitée.

Options de déploiement:

- Docker: containeriser le serveur MCP
- Kubernetes: orchestration à grande échelle
- Serverless: AWS Lambda, Google Cloud Functions
- VM/Serveurs: déploiement traditionnel

12. Sécurité avancée pour MCP

■ Protégez vos serveurs et données

Mesures de sécurité:

- Authentification: tokens JWT ou OAuth
- Chiffrement: TLS/HTTPS obligatoire
- Audit: logger toutes les opérations
- Rate limiting: limiter les requêtes
- Validation: vérifier toutes les entrées

13. Checklist de production

✓ Préparez vos serveurs MCP pour la production

Checklist avant le déploiement:

- ✓ Équilibreurs de charge pour distribuer les requêtes
- ✓ Authentification robuste et sécurisée
- ✓ Logging et monitoring en temps réel
- ✓ Gestion des erreurs et des timeouts
- ✓ Tests de charge et validation performance
- ✓ Documentation complète des APIs
- ✓ Alertes et notifications d'erreur

14. Résumé des sujets avancés

■ Maîtrisez les aspects avancés de MCP

Les sujets avancés de MCP vous permettent de construire des systèmes robustes, sécurisés et scalables. Maîtrisez le sampling, les notifications, les roots, les transports et vous serez prêt pour des déploiements en production.