

Module 6 — MCP et Integrations

Jour 2 — MCP, plugins, CI/CD

The Bearded Bear

Février 2026



Table des matières

Module 6 : MCP et Integrations (1h15)	3
Objectifs	3
1. Le Protocole MCP	3
Definition	3
Architecture	3
Composants MCP	3
Flux d'une requete MCP	4
2. Configuration MCP	4
Via settings.json	4
Via la commande <code>claude mcp add</code>	5
Gestion des serveurs MCP	5
Timeout MCP	5
OAuth pour serveurs sans DCR	5
3. Serveurs MCP Courants	6
Bases de donnees	6
APIs et services	6
Navigateur (Chrome)	7
4. Plugins (via MCP)	7
Installation via MCP	7
LSP Plugins	8
Difference Plugins vs MCP	8
5. Integrations IDE	8
VS Code	8
JetBrains (IntelliJ, PHPStorm, etc.)	9
6. CI/CD	10
Principe	10
GitHub Actions	10
GitLab CI	11
Mode headless avance	11
7. Review Automatisee de PR avec <code>-from-pr</code>	11
Concept	11
Utilisation	11
Auto-link	12
Indicateurs de statut PR	12
Exemple de workflow CI/CD complet	12
8. Securite MCP	13
Risques des serveurs MCP tiers	13
Checklist d'audit MCP	13
Bonnes pratiques	14
Hook de securite MCP	14

Exercice Pratique (15min)	14
Configuration MCP + CI/CD	14
Verification	14
Points Cles a Retenir	15

Module 6 : MCP et Integrations (1h15)

Objectifs

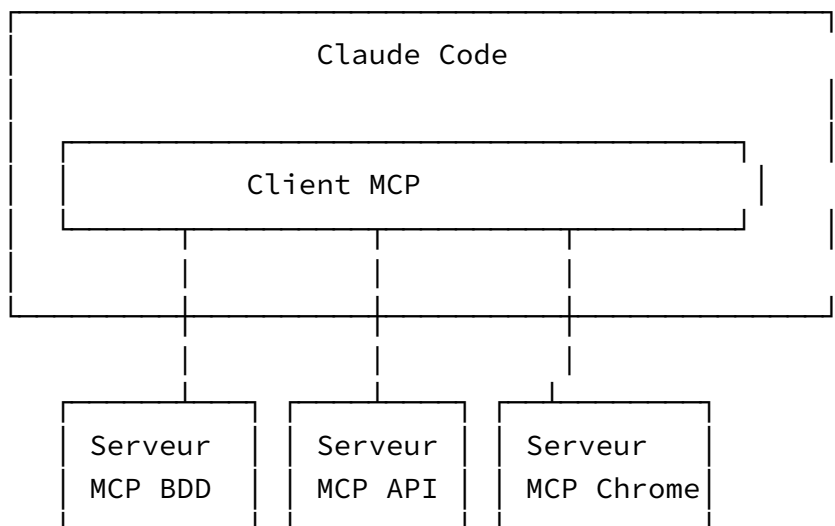
A la fin de ce module, vous serez capable de : - Comprendre le protocole MCP et son architecture - Configurer des serveurs MCP dans Claude Code - Utiliser les serveurs MCP courants (BDD, APIs, navigateur) - Intégrer Claude Code dans les IDEs (VS Code, JetBrains) - Automatiser Claude Code en CI/CD (GitHub Actions, GitLab CI) - Utiliser `--from-pr` pour la review automatisée de Pull Requests - Evaluer et sécuriser les serveurs MCP tiers

1. Le Protocole MCP

Definition

MCP (Model Context Protocol) est un protocole ouvert qui permet à Claude Code de communiquer avec des services externes. Il standardise la manière dont un LLM interagit avec des outils, des ressources et des données.

Architecture



Composants MCP

Composant	Description	Exemple
Serveur MCP	Service qui expose des capacites	Serveur SQLite, serveur GitHub
Outils (Tools)	Actions executables par Claude	query_database, create_issue
Ressources (Resources)	Donnees accessibles en lecture	Schema de BDD, documentation
Prompts	Templates de prompts predefinies	Prompt d'analyse de schema

Flux d'une requete MCP

1. Claude decide d'utiliser un outil MCP
2. Claude Code envoie la requete au serveur MCP
3. Le serveur MCP execute l'action
4. Le resultat est renvoye a Claude Code
5. Claude integre le resultat dans sa reponse

2. Configuration MCP

Via settings.json

La configuration des serveurs MCP se fait dans `.claude/settings.json`:

```
{
  "mcpServers": {
    "sqlite": {
      "command": "npx",
      "args": ["-y", "@anthropic/mcp-server-sqlite", "database.db"]
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@anthropic/mcp-server-github"],
      "env": {
        "GITHUB_TOKEN": "${GITHUB_TOKEN}"
      }
    }
  }
}
```

Via la commande `claude mcp add`

Ajouter un serveur MCP interactivement

```
claude mcp add
```

Ajouter avec des parametres specifiques

```
claude mcp add sqlite -- npx -y @anthropic/mcp-server-sqlite database.db
```

Ajouter avec des variables d'environnement

```
claude mcp add github -- npx -y @anthropic/mcp-server-github \
  --env GITHUB_TOKEN=$GITHUB_TOKEN
```

Gestion des serveurs MCP

Lister les serveurs configures

```
claude mcp list
```

Supprimer un serveur

```
claude mcp remove sqlite
```

Voir le statut en session

```
/mcp
```

Timeout MCP

Par default, les requetes MCP ont un timeout. Configurez-le via la variable d'environnement :

Augmenter le timeout a 60 secondes

```
export MCP_TIMEOUT=60000
```

Ou dans settings.json

```
{
  "env": {
    "MCP_TIMEOUT": "60000"
  }
}
```

OAuth pour serveurs sans DCR

Certains serveurs MCP necessitent une authentication OAuth sans Dynamic Client Registration (DCR). Utilisez les flags `--client-id` et `--client-secret` :

```
claude mcp add my-api \
  --client-id "mon-client-id" \
  --client-secret "mon-client-secret" \
  -- npx -y @example/mcp-server-api
```

3. Serveurs MCP Courants

Bases de donnees

SQLite

```
{
  "mcpServers": {
    "sqlite": {
      "command": "npx",
      "args": ["-y", "@anthropic/mcp-server-sqlite", "./data/app.db"]
    }
  }
}
```

Utilisation :

```
> Montre-moi le schema de la table users
> Combien d'utilisateurs se sont inscrits ce mois-ci ?
> Optimise cette requete qui est lente
```

PostgreSQL

```
{
  "mcpServers": {
    "postgres": {
      "command": "npx",
      "args": ["-y", "@anthropic/mcp-server-postgres"],
      "env": {
        "DATABASE_URL": "postgresql://user:pass@localhost:5432/mydb"
      }
    }
  }
}
```

APIs et services

GitHub

```
{
  "mcpServers": {
    "github": {
      "command": "npx",
      "args": ["-y", "@anthropic/mcp-server-github"],
      "env": {
        "GITHUB_TOKEN": "${GITHUB_TOKEN}"
      }
    }
  }
}
```

Capacites : creer des issues, lire des PRs, rechercher du code, gerer des repositories.

Systeme de fichiers

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": ["-y", "@anthropic/mcp-server-filesystem", "/chemin/autorise"]
    }
  }
}
```

Navigateur (Chrome)

```
{
  "mcpServers": {
    "chrome": {
      "command": "npx",
      "args": ["-y", "@anthropic/mcp-server-chrome"]
    }
  }
}
```

Le serveur Chrome permet a Claude de naviguer sur le web, prendre des screenshots, interagir avec des formulaires et lire le contenu de pages web.

4. Plugins (via MCP)

Note : Les plugins Claude Code sont implementes comme des serveurs MCP specialises. La commande `/plugin` n'est pas confirmee dans toutes les versions. Utilisez la configuration MCP standard pour integrer des serveurs d'analyse de code.

Installation via MCP

```
// Dans .claude/settings.json
{
  "mcpServers": {
    "php-lsp": {
      "command": "node",
      "args": ["path/to/php-lsp-server"]
    }
  }
}
```


LSP Plugins

Les plugins LSP (Language Server Protocol) sont des serveurs MCP qui ajoutent des capacités d'analyse de code spécifiques à un langage. Claude Code peut ainsi accéder aux diagnostics, définitions de types et complétions du LSP.

Stack	Plugin LSP	Capacités
PHP	php-lsp	PHPStan, diagnostics, navigation
Python	pyright-lsp	Type checking, diagnostics
TypeScript	typescript-lsp	tsc diagnostics, navigation
Dart	dart-analyzer	Analyse statique Flutter/Dart
C#	csharp-lsp	OmniSharp, diagnostics .NET

Difference Plugins vs MCP

Aspect	Plugin	MCP
Installation	<code>/plugin install</code>	<code>claude mcp add</code> ou <code>settings.json</code>
Scope	Claude Code spécifique	Protocole universel
Capacités	Pre-intégrées, optimisées	Génériques, extensibles
Maintenance	Par Anthropic / communauté	Par le fournisseur du serveur
Configuration	Minimale	Serveur à configurer

5. Integrations IDE

VS Code

L'extension Claude Code pour VS Code offre une intégration riche :

Fonctionnalités principales

Fonctionnalite	Description	Raccourci
Inline Edit	Modification de code in-place	Cmd+I
Diff View	Vue des modifications proposees	Automatique
Session Picker	Choisir/reprendre une session	Palette de commandes
Python venv	Detection automatique d'environnement	Setting <code>claude-Code.usePythonEnvironment</code>

Configuration VS Code

```
// settings.json (VS Code)
{
  "claudeCode.usePythonEnvironment": true,
  "claudeCode.defaultModel": "sonnet"
}
```

Workflow typique VS Code

1. Ouvrir le fichier a modifier
2. Selectionner le code concerne
3. Cmd+I pour invoquer Claude inline
4. Decrire la modification souhaitee
5. Accepter ou rejeter le diff propose

Variable \$SELECTION Dans VS Code, la variable \$SELECTION dans les slash commands contient le code actuellement selectionne dans l'editeur. Cela permet de creer des commandes contextuelles :

```
<!-- .claude/commands/explain.md -->
Explique ce code en detail :
```

\$SELECTION

Donne :

1. Le role de ce code
2. Les patterns utilises
3. Les ameliorations possibles

JetBrains (IntelliJ, PHPStorm, etc.)

L'integration JetBrains fournit :

Fonctionnalite	Description
Tool Window	Panneau Claude Code integre
Context Sharing	Partage automatique du fichier ouvert
Diff Integration	Vue des modifications dans le diff JetBrains

6. CI/CD

Principe

Claude Code peut fonctionner en **mode headless** dans un pipeline CI/CD. Le flag `-p` (ou `--print`) permet d'envoyer un prompt unique et de recevoir la reponse sans mode interactif.

```
# Mode headless basique
claude -p "Analyse ce code et retourne un rapport JSON"
```

```
# Avec un fichier en input
cat src/main.py | claude -p "Review ce code"
```

GitHub Actions

```
name: Claude Code Review

on:
  pull_request:
    types: [opened, synchronize]

jobs:
  review:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Install Claude Code
        run: npm install -g @anthropic-ai/claude-code

      - name: Run Code Review
        env:
          ANTHROPIC_API_KEY: ${ secrets.ANTHROPIC_API_KEY }
        run: |
          # Obtenir le diff de la PR
```

```
DIFF=$(git diff origin/main...HEAD)

# Envoyer a Claude pour review
echo "$DIFF" | claude -p "Effectue une code review de ce diff. \
  Verifie: securite, performance, lisibilite, tests manquants. \
  Format: JSON avec severity (critical/major/minor) et suggestions."
```

GitLab CI

```
claude-review:
  stage: review
  image: node:20
  script:
    - npm install -g @anthropic-ai/claude-code
    - DIFF=$(git diff origin/main...HEAD)
    - echo "$DIFF" | claude -p "Review ce diff et donne un rapport"
  variables:
    ANTHROPIC_API_KEY: $ANTHROPIC_API_KEY
  only:
    - merge_requests
```

Mode headless avance

```
# Avec un modele specifique
claude -p "Analyse cette PR" --model opus

# Avec un output structure
claude -p "Liste les fichiers modifies au format JSON" --output-format json

# Avec un contexte de projet
claude -p "Verifie la conformite architecture" --project-dir ./src
```

7. Review Automatisee de PR avec -from-pr

Concept

Le flag `--from-pr` permet a Claude Code d'analyser automatiquement une Pull Request GitHub. Claude recoit le contexte complet de la PR (diff, description, commentaires) et peut produire une review structuree.

Utilisation

```
# Review d'une PR par numero
claude --from-pr 123
```

```
# Review d'une PR par URL
```

```
claude --from-pr https://github.com/org/repo/pull/123
```

```
# En mode non-interactif (CI/CD)
```

```
claude -p "Review cette PR pour securite et performance" --from-pr 123
```

Auto-link

Quand Claude Code est utilise avec `--from-pr`, il cree automatiquement un lien entre la session Claude et la PR GitHub. Cela permet de : - Retrouver la session depuis la PR - Voir l'historique des reviews automatisees - Reprendre une review interrompue

Indicateurs de statut PR

Claude Code affiche le statut de la PR dans l'interface :

Indicateur	Signification
approved	PR approuvee
pending	Review en attente
changes_requested	Modifications demandees
draft	PR en brouillon
merged	PR fusionnee

Exemple de workflow CI/CD complet

```
name: Claude PR Review
```

```
on:
```

```
  pull_request:
```

```
    types: [opened, synchronize, ready_for_review]
```

```
jobs:
```

```
  claude-review:
```

```
    runs-on: ubuntu-latest
```

```
    if: github.event.pull_request.draft == false
```

```
    steps:
```

```
      - uses: actions/checkout@v4
```

```
        with:
```

```
          fetch-depth: 0
```

```
      - name: Install Claude Code
```

```
        run: npm install -g @anthropic-ai/claude-code
```

```
- name: Review PR
env:
  ANTHROPIC_API_KEY: ${ secrets.ANTHROPIC_API_KEY }
  GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
run: |
  claude -p "Effectue une code review complete. \
    Verifie: architecture, securite OWASP, performance, tests. \
    Si des problemes critiques sont trouves, indique-les
  ↪ clairement." \
    --from-pr ${ github.event.pull_request.number }
```

8. Securite MCP

Risques des serveurs MCP tiers

Alerte : Des recherches de securite ont identifie des payloads malicieux dans les registres publics de serveurs MCP. Les serveurs MCP tiers non verifies representent un risque significatif.

Risque	Description	Impact
Injection	Parametres MCP manipules pour executer du code	Execution arbitraire
Exfiltration	Serveur qui envoie vos donnees a un tiers	Fuite de code source, secrets
Escalade	Outils qui demandent plus de permissions que necessaire	Acces non autorise
Supply chain	Dependance NPM compromise	Code malicieux installe

Checklist d'audit MCP

Avant d'installer un serveur MCP tiers, verifiez :

- ☐ **Code source disponible** : Le code est-il open source et auditable ?
- ☐ **Auteur verifie** : L'organisation ou l'auteur est-il reconnu ?
- ☐ **Pas d'accès réseau non justifie** : Le serveur a-t-il besoin de faire des requetes HTTP ?
- ☐ **Pas de lecture de fichiers sensibles** : Accede-t-il a `.env`, `.ssh/`, ou des credentials ?
- ☐ **Permissions minimales** : Suit-il le principe du moindre privilege ?
- ☐ **Version pinee** : Utilisez-vous une version fixe (pas `latest`) ?
- ☐ **Changelog et historique** : Y a-t-il un suivi des versions et des correctifs securite ?

Bonnes pratiques

1. PREFERER ecrire ses propres serveurs MCP simples
2. AUDITER le code source avant installation
3. PINER les versions dans la configuration
4. LIMITER les permissions via allowlist d'outils
5. SURVEILLER les logs d'accès MCP
6. ISOLER les serveurs MCP sensibles (sandbox, conteneur)

Hook de securite MCP

Utilisez un hook `PreToolUse` pour surveiller les appels MCP :

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "mcp",
        "command": "echo \"MCP call: $TOOL_NAME\" >> .claude/mcp-audit.log"
      }
    ]
  }
}
```

Exercice Pratique (15min)

Configuration MCP + CI/CD

1. **Configurez un serveur MCP SQLite** (5min)
 - Ajoutez la configuration dans `.claude/settings.json`
 - Testez avec `/mcp` pour verifier la connexion
 - Demandez a Claude de decire le schema de la base
2. **Creez un workflow GitHub Actions** (10min)
 - Creez un fichier `.github/workflows/claude-review.yml`
 - Configurez une review automatique sur les PRs
 - Incluez les verifications de securite et d'architecture

Verification

- ☐ Le serveur MCP SQLite est configure et actif
- ☐ Claude peut interroger la base de donnees
- ☐ Le workflow GitHub Actions est syntaxiquement correct

- ☐ Le workflow utilise `--from-pr` pour le contexte de la PR
-

Points Cles a Retenir

1. **MCP** est un protocole ouvert qui connecte Claude Code a des services externes
 2. **c~~l~~aud~~e~~ mcp add** est le moyen le plus simple d'ajouter un serveur MCP
 3. **Les serveurs courants** couvrent BDD, APIs, navigateur et systeme de fichiers
 4. **Les plugins LSP** ajoutent des capacites d'analyse specifiques par langage
 5. **VS Code et JetBrains** offrent des integrations riches avec Claude Code
 6. **Le mode headless (-p)** permet d'utiliser Claude Code en CI/CD
 7. **--from-pr** connecte Claude Code directement a vos Pull Requests GitHub
 8. **La securite MCP** est critique : auditez toujours les serveurs tiers avant installation
-

Duree : 1h15 **Prochain module :** Module 7 : Multi-Agent et Coordination