

# Cheat Sheet — BMAD & Ralph

BMAD v6, Ralph, QA Recette, Docker

The Bearded Bear

Février 2026



# Cheat Sheet : BMAD v6, Ralph, QA Recette, Docker

## BMAD v6 - Project Management

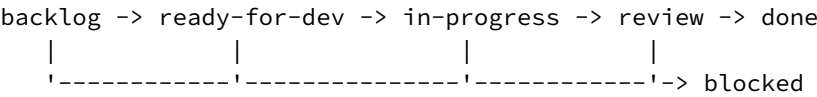
### Initialisation

```
/workflow:init           # Initialiser workflow
/workflow:status        # Statut du projet
```

### Quality Gates (5)

Gate	Seuil	Commande
PRD Gate	>=80%	/gate:validate-prd
Tech Spec Gate	>=90%	/gate:validate-techspec
Backlog Gate	INVEST 6/6	/gate:validate-backlog
Sprint Ready	100%	/gate:validate-sprint
Story DoD	100%	/gate:validate-story

### Status Routing



### Sprint Commands

```
/sprint:next-story      # Prochaine story
/sprint:transition US-1 done # Transition
/sprint:status          # Metriques
/sprint:auto-route      # Auto-routage
/sprint:dev             # Dev TDD
/project:run-sprint     # Sprint complet
```

### Development Tracks

Track	Setup	Best For
Quick Flow	< 5 min	Bug fixes, hotfixes
Standard	< 15 min	New features
Enterprise	< 30 min	Platforms

### Gate Report

```
/gate:report           # Rapport qualite
```

### Project Commands (22 total)

```
/project:run-sprint     # Sprint complet
/project:run-epic       # Epic complet
/project:run-queue      # Executer queue
/project:batch-status   # Statut batch
# +18 commandes gestion backlog, epics, stories
```

## Ralph Wiggum - Boucle IA Continue

### Lancement

```
/common:ralph-run "Implement user authentication"
```

### DoD Validators

Type	Description
command	Run command, check exit code
output_contains	Check output patterns
file_changed	Verify file modified
hook	Custom validation script
human	Manual approval

### Configuration (.ralph.yaml)

```
max_iterations: 50
cost_limit: 5.00
error_threshold: 3
```

### Circuit Breaker

- Max iterations reached -> stop
- Cost limit exceeded -> stop
- Consecutive errors -> stop

## QA Recette - Tests d’Acceptance

### Prerequis

- Chrome extension v1.0.36+
- Claude Code with --chrome or /chrome

### Commandes

```
# Test story / sprint
/qa:recette --scope=story --id=US-001
/qa:recette --scope=sprint --id=Sprint-3

# Dry run (plan only)
/qa:recette --scope=story --id=US-001 --dry-run

# Resume session interrompue
/qa:recette --resume=REC-20260130-143022

# Statut et rapports
/qa:status
/qa:regression
/qa:report

# Corriger bugs
/qa:fix --session=REC-xxx
/qa:fix --session=REC-xxx --dry-run

# Correction TDD
/qa:tdd
```

### Golden Rule

A fixed bug should NEVER reappear.

Auto-generates regression tests for every bug found.

Output Structure

```
.recette/
├── plans/           # Test plans (YAML)
├── sessions/       # Session states
├── regression/     # Regression suite
│   ├── registry.yaml
│   └── tests/
├── metrics/        # Historical data
└── reports/        # Generated reports
```

Docker

Agents (5)

Agent	Expertise
@docker-dockerfile	Dockerfile optimization
@docker-compose	Compose orchestration
@docker-debug	Container troubleshooting
@docker-cicd	CI/CD pipelines
@docker-architect	Docker architecture

Commandes (5)

```
/docker:compose-setup    # Generate docker-compose
/docker:architecture     # Design architecture
/docker:debug            # Diagnose issues
/docker:cicd-pipeline    # Generate CI/CD pipeline
/docker:optimize         # Optimize Docker setup
```

Docker Requirement (CLAUDE.md)

```
# Toujours utiliser Docker
docker compose exec app php bin/console ...
docker compose exec app ./vendor/bin/phpunit
docker compose exec app ./vendor/bin/php-cs-fixer fix
```

Team Commands (Agent Teams)

Commande	Description
/team:audit	Audit multi-tech parallele
/team:sprint	Sprint parallele
/team:security	Revue securite parallele
/team:delivery	Cycle complet sprint

Hook Scripts

Script	Evenement	Description
post-tool-failure.sh	PostToolUseFailure	Auto-recovery apres echec
pre-compact.sh	PreCompact	Backup etat sprint
session-end.sh	SessionEnd	Metriques et cleanup

Configuration

```
{
  "hooks": {
    "PostToolUseFailure": [
      {"command": ".claude/hooks/post-tool-failure.sh"}
    ],
    "PreCompact": [
      {"command": ".claude/hooks/pre-compact.sh"}
    ],
    "SessionEnd": [
      {"command": ".claude/hooks/session-end.sh"}
    ]
  }
}
```

Plan Mode integre

- Les commandes sprint et gate incluent une guidance Plan Mode :
- /sprint:dev,/workflow:implement → **MANDATORY** (plan mode automatique)
  - /gate:validate-\* → **CONDITIONAL** (plan mode si scope large)
  - /sprint:status,/sprint:next-story → pas de plan mode (lecture seule)

BMAD Roles

Les roles BMAD (bmad-master, pm, ba, architect, po, sm, dev, qa, qa-recette, ux) sont integres dans les commandes /workflow:\* et /sprint:\*, pas en agents autonomes.

Project Agents (2)

Agent	Role
@product-owner	Product management (CSPO)
@tech-lead	Technical leadership

Quick Reference

Claude-Craft 7.13.0

Categorie	Nombre
Agents communs	12
Tech reviewers	10

Categorie	Nombre
Agents Docker	5
Agents Coolify	4
Agents projet	2
<b>Total agents</b>	<b>33</b>
Commandes	160
Namespaces	20
Technologies	10
Langues	5
Skills	36

Formation Claude Code 2.1.45 + Claude-Craft 7.13.0 The Bearded CTO - 2026