Utrzymanie i diagnostyka Jenkinsa

Poziom: operacyjny **Dotyczy: Jenkins LTS** **Czas: ~30 min**

Instalacja, która działa miesiącami bez przeglądu, kończy się pełnym dyskiem albo kontrolerem duszącym się pod liczbą równoległych buildów, które sam sobie pozwolił przyjąć. Ta strona zbiera rutynowe czynności utrzymaniowe, których nie widać, dopóki nie zaczną brakować.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
CPU	Central Processing Unit: procesor
GB	Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej
JVM	Java Virtual Machine: maszyna wirtualna, w której działają programy w Javie
LTS	Long Term Support: wersja z długim wsparciem poprawkami
XML	eXtensible Markup Language: format tekstowy do zapisu ustrukturyzowanych danych w znacznikach, tutaj używany przez PowerShell do zapisu zrzutu obiektów poleceniem Export-Clixml

Fiszka

Komendy wykonujemy z CLI bash z uprawnieniami root (sudo) na serwerze kontrolera, chyba że opisano inaczej.

TYLKO ODCZYT

Ograniczenie historii buildów w pipeline: buildDiscarder

```
options {
    buildDiscarder(logRotator(numToKeepStr: '30', artifactNumToKeepStr: '10'))
}
```

Trzymamy 30 ostatnich wpisów historii, ale artefakty tylko z 10 ostatnich — logi buildów zajmują zwykle niewiele, artefakty (obrazy, paczki) potrafią być ciężkie i nie ma sensu trzymać ich tyle samo, co samej historii.

□□ TYLKO ODCZYT

Sprawdzenie łącznej zajętości katalogu: du -sh

```
du -sh /var/lib/jenkins
```

Przykład:

```
41G□/var/lib/jenkins
```

□□ **du** = disk usage: ile miejsca zajmuje katalog

□□ TYLKO ODCZYT

Sprawdzenie, które zadania zajmują najwięcej miejsca: du -sh

```
du -sh /var/lib/jenkins/jobs/*/builds | sort -rh | head -10
```

Zadanie na szczycie listy jest pierwszym kandydatem do zaostrzenia polityki retencji albo do przeniesienia dużych artefaktów poza `JENKINS_HOME` — na zewnętrzny magazyn artefaktów zamiast archiwizowania ich lokalnie przy każdym buildzie.

Przykład:

```
12G□/var/lib/jenkins/jobs/build-obrazu-app/builds
6.1G□/var/lib/jenkins/jobs/testy-integracyjne/builds
2.3G□/var/lib/jenkins/jobs/deploy-staging/builds
```

□□ **du** = disk usage: ile miejsca zajmuje katalog

□□ TYLKO ODCZYT

Podgląd logu przez systemd: journalctl

```
journalctl -u jenkins -f
```

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wyszukanie ostatnich błędów w logu aplikacji: grep

```
grep -i "SEVERE\|ERROR" /var/log/jenkins/jenkins.log | tail -20
```

grep = global regular expression print: szukaj wzorca w tekście

TYLKO ODCZYT

Wylistowanie wszystkich zadań spełniających warunek: Script Console

```
Jenkins.instance.getAllItems(Job.class).each {
    if (it.fullName.contains('legacy')) {
        println it.fullName
    }
}
```

TYLKO ODCZYT

Wymuszenie rozłączenia wszystkich agentów offline: Script Console

```
Jenkins.instance.computers.each { computer ->
    if (computer.isOffline() && !computer.isTemporarilyOffline()) {
        println "Odłączam: ${computer.name}"
        computer.doDoDisconnect(null)
    }
}
```

Skrypt tego typu wykonujemy najpierw z samym `println` bez wywoływania akcji, żeby zobaczyć, na czym operacja by zadziałała, zanim wywołamy jej wersję modyfikującą stan.

□□ TYLKO ODCZYT

Wywołanie bezpiecznego restartu przez URL: curl

```
curl -X POST http://ci01.ad.zielona.example:8080/safeRestart --user
nazwa_usera:HasloUzytkownika
```

□□ **URL** = Uniform Resource Locator: adres strony lub zasobu · **curl** = client URL: pobieranie i wysyłanie przez HTTP z terminala

□□ ZMIENIA SYSTEM

Twardy restart usługi systemd: systemctl restart

```
sudo systemctl restart jenkins
```

Twardy restart przerywa buildy w trakcie wykonywania natychmiast, bez czekania na ich zakończenie — stosujemy go tylko wtedy, gdy kontroler przestał odpowiadać i `safeRestart` nie ma jak się wykonać.

□□ **systemctl** = system control: sterowanie usługami systemd

□□ TYLKO ODCZYT

Sprawdzenie liczby zadań oczekujących przez API: curl

```
curl -s http://ci01.ad.zielona.example:8080/queue/api/json --user
nazwa_usera:HasloUzytkownika | grep -o '"why"' | wc -l
```

□□ **API** = Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą · **curl** = client URL: pobieranie i wysyłanie przez HTTP z terminala

□□ TYLKO ODCZYT

Ustawienie parametrów JVM dla agenta: JVM Options

```
-Xms512m -Xmx1024m
```

W konfiguracji node'a, w sekcji „Advanced", pole „JVM Options" pozwala nadpisać domyślne ustawienia sterty procesu agenta. Ustawiamy je, gdy agent regularnie wykonuje buildy zużywające dużo pamięci (np. duże projekty Javy czy Node.js) i domyślna wartość

maksymalnej sterty prowadzi do `OutOfMemoryError` w trakcie kompilacji, mimo że sama maszyna ma zapas pamięci fizycznej.

JVM = Java Virtual Machine: maszyna wirtualna, w której działają programy w Javie

TYLKO ODCZYT

Sprawdzenie wtyczek z dostępną aktualizacją: curl

```
curl -s http://ci01.ad.zielona.example:8080/pluginManager/api/json?depth=1 --user
nazwa_usera:HasloUzytkownika | grep -o '"shortName":"[^"]*","hasUpdate":true' | head -20
```

curl = client URL: pobieranie i wysyłanie przez HTTP z terminala

ZMIENIA SYSTEM

Cofnięcie wtyczki do poprzedniej wersji: ręczne przywrócenie

```
sudo systemctl stop jenkins
sudo mv /var/lib/jenkins/plugins/NazwaWtyczki.jpi.bak
/var/lib/jenkins/plugins/NazwaWtyczki.jpi
sudo systemctl start jenkins
```

Jenkins przechowuje jedną poprzednią wersję każdej zainstalowanej wtyczki w katalogu `plugins/` pod rozszerzeniem `.jpi.bak`.

systemctl = system control: sterowanie usługami systemd · **mv** = move: przenieś albo zmień nazwę

Czyszczenie starych buildów

Pojęcie: polityka retencji (Discard Old Builds). Ustawienie na poziomie konfiguracji zadania, określające, ile ostatnich buildów (albo ile dni historii) Jenkins ma zachować, zanim zacznie automatycznie kasować najstarsze wraz z ich workspace i artefaktami. Bez tej polityki historia zadania rośnie w nieskończoność, a wraz z nią zajętość `JENKINS_HOME`.

Logi

> **Pojęcie: dwa miejsca logowania kontrolera.** Instalacja z pakietu systemowego pisze log aplikacji w dwóch miejscach naraz — plik `/var/log/jenkins/jenkins.log` (jeśli tak skonfigurowano logowanie) oraz standardowy dziennik systemd, do którego trafia to samo wyjście, gdy usługa jest uruchamiana bez przekierowania do osobnego pliku.

Skrypt konsoli Groovy do zadań masowych

> **Pojęcie: Script Console.** Konsola dostępna pod **Manage Jenkins → Script Console**, wykonująca dowolny skrypt Groovy z pełnym dostępem do wewnętrznego API Jenkinsa — najszybszy sposób na operację masową, której nie da się wykonać przez UI zadanie po zadaniu, ale też najszybszy sposób na uszkodzenie instalacji przy błędzie w skrypcie. Konsola wymaga uprawnień administratora i nie przechodzi przez zwykły mechanizm Script Approval — kod uruchamia się natychmiast.

Restart bezpieczny kontra twardy

> **Pojęcie: safeRestart.** Restart poprzedzony wstrzymaniem przyjmowania nowych buildów i odczekaniem na zakończenie tych już trwających — wywoływany z UI (**Manage Jenkins → Reload Configuration from Disk** ma inne zastosowanie; tu chodzi o przycisk restartu) albo bezpośrednio adresem `/safeRestart`. To jest domyślny, bezpieczny sposób restartowania kontrolera po instalacji wtyczki albo zmianie konfiguracji.

Kolejka zadań

>

Pojęcie: kolejka builda (Build Queue). Widoczna w UI lista zadań czekających na wolnego agenta pasującego etykietą — zadanie utknie w kolejce, jeśli żaden agent nie ma wymaganej etykiety, wszyscy pasujący agenci są zajęci, albo zadanie czeka na zwolnienie zasobu zablokowanego przez `lock`. Długo rosnąca kolejka to najpewniejszy sygnał, że liczba agentów przestała nadążać za liczbą zadań.

Liczba równoległych buildów

Pojęcie: executor. Slot na kontrolerze albo agencie, w którym może wykonywać się dokładnie jeden build naraz — liczba executorów danego node'a to górny limit równoległości na tej konkretnej maszynie, ustawiany w polu „# of executors" w konfiguracji node'a (albo kontrolera, w **Manage Jenkins → System**, jeśli kontroler też przyjmuje buildy). Wartość tego pola nie powinna przekraczać liczby rdzeni CPU maszyny — więcej executorów niż rdzeni oznacza, że buildy zaczynają walczyć o ten sam procesor zamiast wykonywać się faktycznie równolegle, co w praktyce tylko wydłuża każdy z nich.

Wtyczki — aktualizacja i cofanie

Aktualizacja wtyczki z poziomu UI

Aktualizację pojedynczej wtyczki wykonujemy z Manage Jenkins → Plugins → Updates, zaznaczając wybraną pozycję i klikając „Download now and install after restart" — Jenkins pobiera nową wersję, ale nie podmienia jej na gorąco, tylko przy najbliższym restarcie (bezpiecznym albo twardym).

Cofnięcie wtyczki do poprzedniej wersji

> Cofnięcie wtyczki bez sprawdzenia, czy inne zainstalowane wtyczki nie zaczęły już wymagać nowszej wersji jako zależności, kończy się czasem błędem ładowania przy starcie — warto to zrobić od razu w środowisku testowym, nie na produkcyjnym kontrolerze.

Kopia zapasowa JENKINS_HOME

Kopia całego katalogu `JENKINS_HOME` to jedyny sposób, żeby po awarii dysku albo nieudanej aktualizacji wrócić do stanu sprzed problemu. Zawartość tego katalogu opisuje tabela na stronie Architektura Jenkinsa: kontroler, agenci, workspace: bez podkatalogu `secrets/` (klucz szyfrujący magazyn poświadczeń) kopia jest bezużyteczna do odtworzenia haseł zapisanych w Jenkinsie, więc kopiujemy cały katalog naraz, nie wybrane pliki.



Skrypt kopii co noc pakuje JENKINS_HOME z ci01 i wysyła archiwum na bkp01 (kliknięcie otwiera pełny rozmiar)

Kopia przez tar, przy zatrzymanym kontrolerze

```
sudo systemctl stop jenkins
sudo tar czf /SciezkaDoPliku/jenkins_home_$(date +%F).tar.gz -C /var/lib/jenkins .
sudo systemctl start jenkins
```

Zatrzymanie usługi na czas kopii gwarantuje spójny stan plików XML w trakcie zapisu, kosztem kilku minut przerwy w przyjmowaniu buildów. Przy instalacji w Dockerze, opisanej na stronie Instalacja Jenkinsa w Dockerze, ten sam efekt daje `docker compose stop jenkins` przed `tar`em i `docker compose start jenkins` po nim, z katalogiem `/docker/apps/jenkins` w miejsce `/var/lib/jenkins`.

Harmonogram

Wybieramy cron: zadania cykliczne albo systemd timery: OnCalendar zamiast crona i spinamy nim obie komendy z kroku wyżej w jeden skrypt uruchamiany co noc, poza godzinami szczytu buildów. Przykładowy wpis crona:

```
crontab -e
```

Przykład:

```
0 3 * * * /var/lib/jenkins-skrypty/kopia-nocna.sh >> /var/log/jenkins-kopia.log 2>&1
```

Kopie trafiają na `bkp01`, do katalogu `/backup/jenkins/`, tym samym mechanizmem co reszta serwerów opisanych w Świat przykładów: Zielona:

```
scp /SciezkaDoPliku/jenkins_home_2026-09-27.tar.gz nazwa_usera@bkp01:/backup/jenkins/
```

Test odtworzenia

Raz na kwartał odtwarzamy najnowszą kopię na osobnej, testowej instalacji tej samej wersji Jenkinsa: zatrzymujemy testowego Jenkinsa, czyścimy jego `JENKINS_HOME`, rozpakowujemy archiwum na jego miejsce, uruchamiamy ponownie i sprawdzamy, że zadania i poświadczenia są na miejscu. Kopia, której nikt nigdy nie spróbował odtworzyć, jest tylko przypuszczeniem, że działa.

> Kopia wykonana bezpośrednio przed aktualizacją, opisana na stronie Instalacja i aktualizacja Jenkinsa w sekcji Wycofanie zmian, to osobny, dodatkowy zrzut, świeższy niż ostatnia kopia nocna, wykonywany tuż przed zmianą wersji, nie zamiast harmonogramu z tej strony.

Częste błędy

Ustawienie liczby executorów kontrolera wyższej niż liczba rdzeni „na wszelki wypadek". Skutek jest odwrotny do zamierzonego — więcej buildów walczy o ten sam procesor, każdy z nich wykonuje się wolniej, a łączny czas oczekiwania w kolejce rośnie zamiast maleć.

Uruchamianie modyfikującego skryptu w Script Console bez wcześniejszego przetestowania wersji tylko do odczytu. Skrypt Groovy w tej konsoli ma pełen dostęp do wewnętrznego stanu Jenkinsa — błąd w warunku pętli potrafi rozłączyć albo skasować więcej, niż zamierzano.

Twardy restart jako pierwsza reakcja na wolno działający kontroler. Przerywa wszystkie trwające buildy natychmiast — `safeRestart` w niemal każdej sytuacji poza całkowitym zawieszeniem procesu jest właściwym pierwszym krokiem.

Ćwiczenia

> Zadania sprawdzające.

1. Zadanie nie ma ustawionej polityki retencji i jego katalog `builds/` zajął 12 GB. Piszemy fragment pipeline'u ustawiający limit 20 ostatnich buildów i 5 ostatnich zestawów artefaktów.
2. Wyjaśniamy różnicę między `safeRestart` a `systemctl restart jenkins` i podajemy sytuację, w której tylko ten drugi ma sens.
3. Kontroler ma 8 rdzeni CPU i ustawioną liczbę executorów na 24. Opisujemy, jaki efekt to prawdopodobnie ma na czas trwania pojedynczego builda.
4. Wskazujemy, gdzie szukać poprzedniej wersji wtyczki po nieudanej aktualizacji i czego trzeba dopilnować przed jej ręcznym przywróceniem.

Powiązane: Architektura Jenkinsa: kontroler, agenci, workspace · Instalacja i aktualizacja Jenkinsa · Agenci: dodanie, diagnostyka, utrzymanie

---

Revision #36
Created 2026-09-25 07:39:56 UTC by Tomasz Konieczny
Updated 2026-10-26 14:15:25 UTC by Tomasz Konieczny