



De usuário a contribuidor

Entenda os workflows da comunidade Postgres

Israel Barth Rubio,
Senior Staff SDE
04/09/2025

A motivação



Por que contribuir para o Postgres parece tão difícil?

- Processos rigorosos
 - Garantir qualidade
 - Comunidade exigente
- Processos únicos
 - Não utiliza as populares pull requests, issues e discussions
- Uso de listas de e-mail

Por que você deveria contribuir?

- Mantido 100% pela comunidade
- Um dos maiores projetos open source do mundo
- Impacto real
- Melhorar a ferramenta que você usa
- Reconhecimento
 - <https://www.postgresql.org/community/contributors/>
 - <https://www.postgresql.org/docs/17/release-17.html#RELEASE-17-ACKNOWLEDGEMENTS>

Por que você deveria contribuir?

- Ganhar moedas comemorativas



<https://wiki.postgresql.org/images/c/cd/Pg17-challenge-coin.jpg>

O ecossistema de desenvolvimento

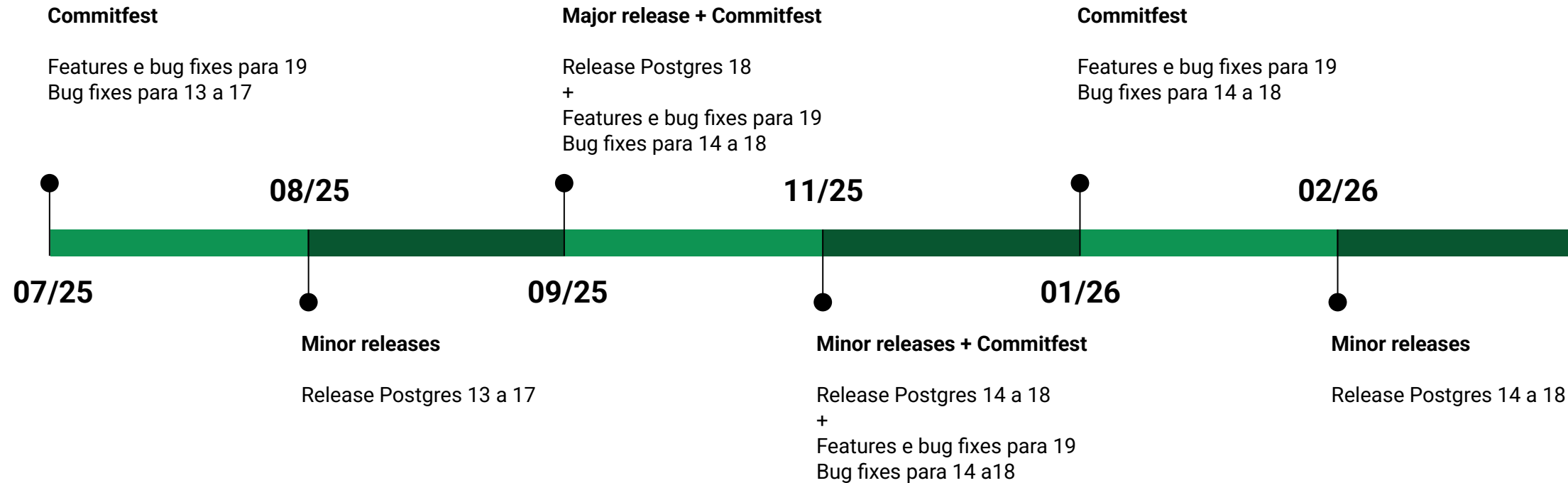


Estrutura de desenvolvimento

- Nova versão (major version) uma vez ao ano
- Versão corretiva (minor version) uma vez a cada 3 meses
- 5 últimas major versions são suportadas
- Contribuições agrupadas em ciclos de 1 mês (Commitfest) a cada 2 meses
 - Discussão, revisão e teste
 - Aprovação e commit
 - Qualquer pessoa pode submeter ou revisar patches
 - O ideal é que cada autor revise pelo menos um patch de outra pessoa

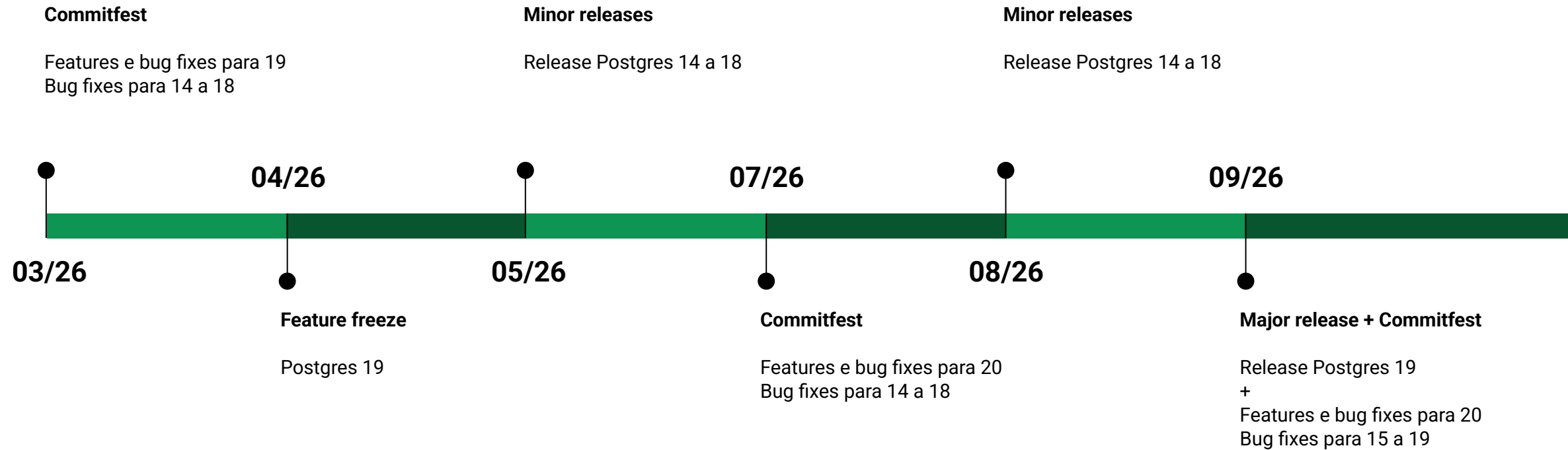
Estrutura de desenvolvimento

- Utilizando Postgres 19 como exemplo



Estrutura de desenvolvimento

- Utilizando Postgres 19 como exemplo



Commitfest (app)

Commitfest PG19-2 (2025-09-01 - 2025-09-30)

Search/filter

Shortcuts ▾

New patch












Status summary: Needs review: 113. Waiting on Author: 22. Ready for Committer: 19. Committed: 6. Moved to next CF: 15. Rejected: 1. Withdrawn: 2. Total: 178.

Active patches

Patch	ID	Status	Tags	Ver	CI status	Stats	Author	Reviewers	Committer	Num cfs	Latest mail
Bug Fixes											
pg_ctl start may return 0 even if the postmaster has been already started on Windows	4573	Waiting on Author			Needs rebase!		Kyotaro Horiguchi (horiguti)	Hayato Kuroda (ha-kun), Yasir Shah (yasir.hussain.shah@gmail.com)		9	4 months ago
Fix rare recovery shutdown hang	4884	Waiting on Author			Needs rebase!		Thomas Munro (maddice)	Noah Misch (nmisch), Martijn Walleet (mwallet)		6	8 months ago
WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts	4986	Waiting on Author			10/10	+6-2	Matthias van de Meent (mmeent)	Akshat Jaimini (akshatj)		6	8 months ago
Historic snapshot doesn't track txns committed in BUILDING_SNAPSHOT state	5029	Needs review			Needs rebase!		ChangAo Chen (chenchangao)			6	1 month ago
Incorrect error message for cancellation triggered by statement_timeout & flaky test	5070	Needs review	Bugfix Flakyfix	stable	10/10	+16-5	Jelte Fennema-Nio (jeltef)			7	5 months ago
Incremental Sort Cost Estimation Instability	5093	Needs review		18	Needs rebase!		Andrei Lepikhov (lepikhov)			6	2 months ago

Commitfest (app)

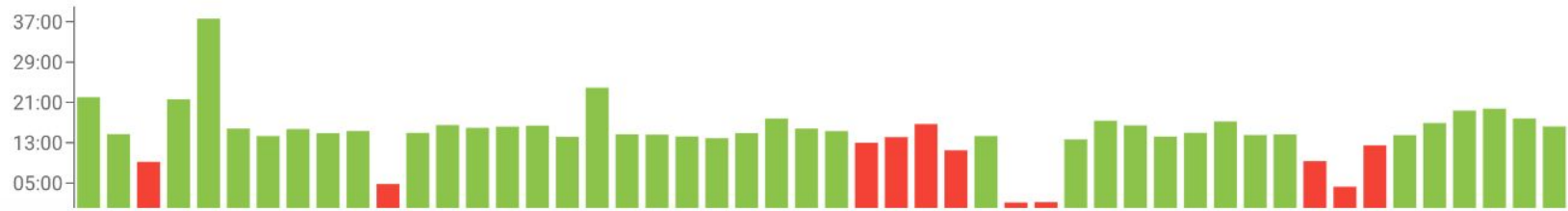
WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts

Edit	Comment/Review ▾	Change Status ▾
ID	4986	
Title	WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts	
CI (CFBot)	<div> Summary          </div> <div>Copy git checkout commands</div>	
Stats (from CFBot)	Patch version: v1, Patch count: 1, First patch: +6-2, All patches: +6-2	
Topic	Bug Fixes	
Tags		
Created	2024-05-13 12:32:12	
Last modified	2025-07-24 15:44:15 (2 weeks ago)	
Latest email	2024-12-10 06:25:16 (8 months ago)	
Status	<div>PG19-2 (2025-09-01 - 2025-09-30): Waiting on Author</div> <div>2025-03 (2025-03-01 - 2025-04-08): Moved to different CF</div> <div>.....</div>	

Commitfest (app)

pgsql-cfbot > postgresql > cf/4986

Duration Chart



Builds

					Repository	Owner	Branch	
completed	[CF 4986] v1 - WAL_LOG CREATE DATABASE strategy broken for ...	0672bc7	1:48:24 AM 8/11/2025		postgresql	pgsql-cfbot	cf/4986	16:13
completed	[CF 4986] v1 - WAL_LOG CREATE DATABASE strategy broken for ...	526d9fb	4:01:26 PM 8/9/2025		postgresql	pgsql-cfbot	cf/4986	17:50

Commitfest (app)













postgresql-cfbot > postgresql > cf/4986 > Build for 0672bc7

 Created 1 day ago  Finished in 15:38



[CF 4986] v1 - WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts

Commit [0672bc7](#) on branch [cf/4986](#).

			
TASKS (10)		HOOKS (0)	
	00:46		SanityCheck
	07:57		FreeBSD - Meson
	00:00		NetBSD - Meson
	00:00		OpenBSD - Meson
	10:09		Linux - Debian Bookworm - Autoconf

Commitfest (app)

Authors	Matthias van de Meent (mmeent)																				
Reviewers	Akshat Jaimini (akshatj)		<button>Become reviewer</button>																		
Committer																					
Links																					
Emails	<div><div>WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts ✕</div><div>First at 2024-05-13 12:31:41 by Matthias van de Meent <boekewurm+postgres at gmail.com> Latest at 2024-12-10 06:25:16 by Michael Paquier <michael at paquier.xyz> Latest attachment (v1-0001-Fix-logging-of-non-standard-pages-in-RelationCopy.patch) at 2024-05-13 12:31:41 from Matthias van de Meent <boekewurm+postgres at gmail.com> <div>+</div></div></div> <div><div>Attach thread</div></div>																				
History	<table><tr><th>When</th><th>Who</th><th>What</th></tr><tr><td>2025-07-24 15:44:15</td><td>Álvaro Herrera (alvherre)</td><td>Moved from CF 2025-03 to CF PG19-2</td></tr><tr><td>2025-02-12 16:04:05</td><td>Álvaro Herrera (alvherre)</td><td>Closed in commitfest 2025-01 with status: Moved to next CF</td></tr><tr><td>2024-12-10 06:25:13</td><td>Michael Paquier (michael-kun)</td><td>Closed in commitfest 2024-11 with status: Moved to next CF</td></tr><tr><td>2024-10-04 17:52:29</td><td>Andrey Borodin (x4m)</td><td>Closed in commitfest 2024-09 with status: Moved to next CF</td></tr><tr><td>2024-10-04 17:52:24</td><td>Andrey Borodin (x4m)</td><td>New status: Waiting on Author</td></tr></table>			When	Who	What	2025-07-24 15:44:15	Álvaro Herrera (alvherre)	Moved from CF 2025-03 to CF PG19-2	2025-02-12 16:04:05	Álvaro Herrera (alvherre)	Closed in commitfest 2025-01 with status: Moved to next CF	2024-12-10 06:25:13	Michael Paquier (michael-kun)	Closed in commitfest 2024-11 with status: Moved to next CF	2024-10-04 17:52:29	Andrey Borodin (x4m)	Closed in commitfest 2024-09 with status: Moved to next CF	2024-10-04 17:52:24	Andrey Borodin (x4m)	New status: Waiting on Author
When	Who	What																			
2025-07-24 15:44:15	Álvaro Herrera (alvherre)	Moved from CF 2025-03 to CF PG19-2																			
2025-02-12 16:04:05	Álvaro Herrera (alvherre)	Closed in commitfest 2025-01 with status: Moved to next CF																			
2024-12-10 06:25:13	Michael Paquier (michael-kun)	Closed in commitfest 2024-11 with status: Moved to next CF																			
2024-10-04 17:52:29	Andrey Borodin (x4m)	Closed in commitfest 2024-09 with status: Moved to next CF																			
2024-10-04 17:52:24	Andrey Borodin (x4m)	New status: Waiting on Author																			

Lista de e-mail

Re: WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts

From: Michael Paquier <michael(at)paquier(dot)xyz>
To: Akshat Jaimini <destrex271(at)gmail(dot)com>
Cc: pgsql-hackers(at)lists(dot)postgresql(dot)org, Matthias van de Meent <boekewurm+postgres(at)gmail(dot)com>
Subject: Re: WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts
Date: 2024-12-10 06:25:16
Message-ID: Z1fezFJJwBxtnkD@paquier.xyz
Views: [Whole Thread](#) | [Raw Message](#) | [Download mbox](#) | [Resend email](#)
Thread: 2024-12-10 06:25:16 from Michael Paquier <michael(at)paquier(dot)xyz> 
Lists: [pgsql-hackers](#)

On Sat, Sep 14, 2024 at 06:57:21PM +0000, Akshat Jaimini wrote:
> Quick question, are there any more revisions left to be done on this
> patch from the previous feedback?

This patch is still listed in the CF app waiting on author with what looks like Robert and Tom objecting to it, because it makes all callers of CreateAndCopyRelationData() much more expensive in terms of WAL generated when copying these files as the holes of the pages would be included. This would also make the compression of the FPWs more expensive.

One potential way to move forward would be to spread the knowledge of page_std when logging a new page higher in the callers, then allow dbcommands.c to be smarter about that?
--
Michael

In response to

- [Re: WAL_LOG CREATE DATABASE strategy broken for non-standard page layouts](#) at 2024-09-14 18:57:21 from Akshat Jaimini

Browse pgsql-hackers by date

	From	Date	Subject
Next Message	Michael Paquier	2024-12-10 06:27:29	Re: long-standing data loss bug in initial sync of logical replication
Previous Message	Dilip Kumar	2024-12-10 06:18:44	Re: Skip collecting decoded changes of already-aborted transactions

Preparando seu ambiente



Pré-requisitos

- **git**: Clonar repositórios, criar e aplicar patches
- **gcc**: Compilar Postgres
- **perl**: Executar testes
- **flex** e **bison**: Gerar os analisadores léxicos e sintáticos da linguagem SQL
- **autoconf** + **make** ou **meson** + **ninja**: Orquestrar configuração, compilação e testes
 - **autoconf** + **make**: Oficial, funciona em sistemas baseados em Unix
 - **meson** + **ninja**: Experimental, funciona em sistemas baseados em Unix e Windows

Preparando o código fonte

- Repositório oficial em **git://git.postgresql.org/git/postgresql.git**
- Mirror em <https://github.com/postgres/postgres>
 - Crie um fork na sua conta do GitHub
 - Facilita integração com CI
 - Clone o seu fork: `git clone git@github.com:<username>/postgres.git`

Testes automatizados no seu fork: Cirrus CI

- É a mesma ferramenta de CI utilizada pelo app Commitfest
- Como configurar
 - Encontre o app Cirrus CI no GitHub Marketplace
 - Instale esse app gratuitamente
 - Habilite acesso do app ao seu fork do Postgres
- Como usar
 - A cada commit ou PR em seu fork, um novo job será disparado no Cirrus CI
 - Você pode conferir os jobs e seus artefatos em <https://cirrus-ci.com/github/<username>/postgres/>
- Instruções mais completas em **src/tools/ci/README**

Compilando o código fonte (make)

- Configurar a compilação, por exemplo: `./configure --prefix=/home/vagrant/pgsql --with-openssl --enable-debug --enable-tap-tests`
 - O configure possui diversas flags e você pode customizar
 - Você precisa satisfazer as dependências do que você configurar. No exemplo, assumindo RHEL 9, pelo menos:
 - **openssl-devel** e **perl(IPC::Run)**
 - Mais informações em <https://www.postgresql.org/docs/current/install-requirements.html>
 - Se quiser desfazer a configuração no futuro: `make distclean`
- Compilar: `make -j $(nproc)`
 - Cria vários arquivos **.o** e arquivos executáveis dentro de **src**
 - Se quiser remover os artefatos de compilação no futuro: `make clean`

Compilando o código fonte (make)

- Instalar: `make install`
 - Instala executáveis e bibliotecas dentro do seu **--prefix**
 - Se quiser desinstalar no futuro: `make uninstall`

Compilando o código fonte (meson)

Ação	make	meson
Configurar	<pre>./configure \ --prefix=/home/vagrant/pgsql \ \ --with-openssl \ --enable-debug \ --enable-tap-tests</pre>	<pre>meson setup builddir \ --prefix=/home/vagrant/pgsql \ -Dssl=openssl \ -Ddebug=true \ -Dtap_tests=enabled</pre>
Limpar tudo	<pre>make distclean</pre>	<pre>rm -rf builddir</pre>
Compilar	<pre>make -j \$(nproc)</pre>	<pre>ninja -C builddir</pre>
Limpar compilação	<pre>make clean</pre>	<pre>ninja -C builddir clean</pre>
Instalar	<pre>make install</pre>	<pre>ninja -C builddir install</pre>
Desinstalar	<pre>make uninstall</pre>	<pre>ninja -C builddir uninstall</pre>

Executando os testes localmente (make)

- Para executar a suite completa: `make check-world`
- Para executar testes específicos: `make -C <caminho> check`
 - Por exemplo, testes de **pg_basebackup** : `make -C src/bin/pg_basebackup check`
- É uma facilidade para testes locais, mas não substitui o Cirrus CI:
 - CI testa em diferentes ambientes, como Windows e Linux
 - CI testa com diferentes ferramentas, como **make** e **meson**
 - CI testa com configurações mais extremas no **./configure**
 - Por exemplo: Segmentos de arquivos de dados com **48kB** ao invés de **1GB**

Executando os testes localmente (meson)

- Observação: comandos **meson** para teste precisam ser executados dentro de **builddir**

Ação	make	meson
Testar tudo	<code>make check-world</code>	<code>meson test</code>
Testar algo	<code>make -C src/bin/pg_basebackup check</code>	<code>meson test \ --suite setup \ --suite pg_basebackup</code>

Compilando a documentação

- No diretório **doc/src/sgml**
- Instalar dependências. Assumindo RHEL 9: **docbook-dtds** , **docbook-style-xsl** e **libxslt**
- Compilar: `make STYLE=website html`
 - Artefatos gerados na pasta **html**
- Mais informações em <https://www.postgresql.org/docs/current/docguide.html>

Código fonte



Overview do repositório de código

- **contrib** : Ferramentas e features auxiliares que não fazem parte do core, por exemplo **pg_stat_statements**
- **doc** : Toda a documentação do Postgres (formato SGML)
- **src** : Core do Postgres
 - **backend** : Postgres server (comandos, parser, planner, executor, background workers, etc.)
 - **bin** : Clients como **psql**, **pg_dump** e **pg_basebackup**
 - **common** : Código compartilhado entre frontend e backend
 - **include** : Arquivos header (**.h**)
 - **interfaces** : Definição de interfaces de comunicação, por exemplo protocolo **libpq**
 - **port** : Hacks para portabilidade de código, por exemplo para lidar com hard-links no Windows
 - **tools** : Ferramentas de apoio ao desenvolvedor, por exemplo **pg_indent**

Convenções de código

- Indentação com tab de 4 espaços, preservando o tab
- Linhas legíveis de até 80 caracteres, se possível
- Comentários com `/* ... */`. Não usar `//`
- Siga as convenções BSD de formatação. Use o mesmo estilo local de cada arquivo sendo alterado

```
/* Comentário de uma linha. */  
if (condicao)  
    FacaAlgo();  
/*  
 * Comentário de  
 * múltiplas linhas.  
 */  
else  
{  
    FacaOutraCoisa();  
    FacaOutraCoisaMais();  
}
```



```
// Comentário de uma linha.  
if (condicao) {  
    FacaAlgo();  
}  
// Comentário de  
// múltiplas linhas.  
else {  
    FacaOutraCoisa();  
    FacaOutraCoisaMais();  
}
```



Listas de e-mail



Convenções de comunicação

- Listas de e-mail
 - **pgsql-general** : Discussões gerais e dúvidas de usuários
 - **pgsql-bugs** : Onde os bugs reportados via <http://www.postgresql.org/support/submitbug> são postados
 - **pgsql-docs** : Discussões a respeito da documentação do Postgres
 - **pgsql-hackers** : Discussões a respeito de desenvolvimento (novas funcionalidades, patches em andamento, bug fixes, etc.)
 - Lista completa e como se inscrever em <https://www.postgresql.org/list/>

Convenções de comunicação

- Orientações gerais
 - Ataques pessoais não são tolerados
 - Seja exigente com o que você escreve e tolerante com o que você lê
 - Tenha paciência, as respostas podem ser rápidas ou lentas
 - Não se assuste com a comunicação técnica e “agressiva” da comunidade
 - Use a opção “responder para todos” ao responder e-mails
 - Não escreva e-mails usando **top-posting**
 - Código de conduta em <https://www.postgresql.org/about/policies/coc/>

Convenções de comunicação

```
> A performance será ruim na condição X.  
> Sugiro fazer Y para melhorar a  
performance  
> nesse caso.
```

Segue em anexo o novo patch com a
alteração sugerida.



Segue em anexo o novo patch com a
alteração sugerida.

```
----- Original Message -----  
From: Reviewer <reviewer@example.com>  
Sent: Tuesday, August 12, 2025 10:01 AM  
To: Author <author@example.com>  
Subject: RE: Patch PG 19
```



A performance será ruim na condição X.
Sugiro fazer Y para melhorar a
performance nesse caso.

Criando e submetendo patches



Como submeter seu patch

- Pense em algo novo que você gostaria de implementar ou selecione algo em <https://wiki.postgresql.org/wiki/Todo>
 - Verifique se sua ideia não está listada em https://wiki.postgresql.org/wiki/Not_Worth_Doing
 - Procure por possíveis discussões existentes nas listas de e-mail a respeito do assunto selecionado
 - Se não houver, inicie uma discussão na lista **pgsql-hackers**
- Caso tenha um feedback positivo na lista de e-mails, inicie suas alterações
 - `git checkout master; git pull; git checkout -b dev/minha-feature`
 - Realize suas alterações no código e/ou testes e/ou documentação, conforme o escopo do seu patch
 - Evite patches muito grandes. São mais difíceis de revisar e serem aceitos
 - Certifique-se de que os testes estão passando, tanto localmente quanto no seu Cirrus CI

Como submeter seu patch

- Uma vez que seu patch esteja pronto, submeta para revisão
 - `git format-patch -v 1 -o /tmp master..dev/minha-feature`
 - Envie o(s) arquivo(s) **.patch** na lista de e-mail
 - Crie uma entrada na próxima Commitfest

Como submeter seu patch

New patch

Specify thread
msgid:

Find thread

Description:

Topic:

Tags:

Wikilink:

Gitlink:

Target version:

Authors:

Israel Barth (barthisrael) ✕

Enter part of name to see list

Create patch

Como submeter seu patch

- Acompanhe as revisões submetidas na lista e alterações de status no Commitfest
 - Responda os e-mails e altere o status conforme necessário
 - Se precisar enviar uma nova versão do patch:
 - Faça rebase para garantir estar na última posição do **master**: `git rebase master`
 - Incremente o **-v** no **git format-patch** para gerar um ID único para cada versão do patch
- Ao final das revisões, se o patch for aceito por um committer, ele será incluído no repositório principal
 - Uma vez incluído, fique de olho na **buildfarm** para verificar se não houve problemas com testes ou portabilidade
 - **buildfarm** é um grupo de máquinas variadas que estão constantemente compilando e testando o Postgres
 - https://buildfarm.postgresql.org/cgi-bin/show_status.pl
- Mais informações em https://wiki.postgresql.org/wiki/Submitting_a_Patch

Revisando patches



Como revisar um patch

- Procure algum patch para revisar na Commitfest
 - Clique no botão **Become Reviewer**
- Verifique que todos os testes do patch passam no Cirrus CI do app Commitfest
- Faça o download do(s) arquivo(s) **.patch** e aplique eles:
 - Crie uma branch para testar: `git checkout master; git pull; git checkout -b review/patch-x`
 - Aplique o(s) patch(es): `git apply <arquivo>.patch`

Como revisar um patch

- Faça uma revisão sob diferentes perspectivas, conforme suas habilidades:
 - **Submissão** : Patch aplica sem erros? Inclui modificações necessárias em testes e documentação?
 - **Usabilidade** : Implementa o que é proposto e da forma esperada? Há problemas? Cobre o que é esperado?
 - **Performance** : Impacta performance de uso ou testes? Melhora performance, caso seja um patch para isso?
 - **Código** : Segue as convenções? Respeita portabilidade? Produz warnings na compilação? Crasha na execução?

Como revisar um patch

- Submeta suas revisões na lista de e-mail e atualize o status no Commitfest de acordo
 - Mantenha esse ciclo até terminar de fazer todas as suas ponderações
- Uma vez que tenha consenso sobre o patch com todos os revisores, aguarde a revisão final de um committer
 - Um committer fará uma última revisão antes de aplicar o(s) patch(es) sugerido(s)
- Mais informações em https://wiki.postgresql.org/wiki/Reviewing_a_Patch

Informações auxiliares



Como funcionam as correções de bugs

- Bugs geralmente são reportados via **pgsql-bugs** ou **pgsql-hackers**
- O patch geralmente é aplicado em todos os branches estáveis que contém o problema, a menos que:
 - Comprometa estabilidade
 - Necessite de alteração de alguma API
 - Necessite de outros patches

Como funcionam as correções de segurança

- Gerenciadas por um time específico de segurança: security@postgresql.org
- Correções são discutidas em privado
- Patches são aplicados próximo à data de lançamento
- Notas de lançamento são feitas, pacotes criados, e então o anúncio público
- Mais informações em <https://www.postgresql.org/support/security/>

Conclusão



Você está pronto para contribuir

- Comece com revisões ou submetendo patches pequenos
 - São uma excelente forma de contribuir e aprender ao mesmo tempo
 - Até mesmo uma pequena correção ou melhoria na documentação pode ser uma ótima oportunidade
- Seu patch não chamou atenção de nenhum committer? Não desista:
 - Tente dar publicidade ao seu patch através de conferências
 - Tente discutir e “vender seu patch” para um committer durante as conferências
 - Talvez seu patch esteja grande e você somente precisa dividir o problema em pedaços menores
- Se desejar se aprofundar mais:
 - https://wiki.postgresql.org/wiki/So_you_want_to_be_a_developer
 - https://wiki.postgresql.org/wiki/Development_information
 - https://wiki.postgresql.org/wiki/Developer_FAQ

Perguntas?

barthisrael@gmail.com

